The Comparison of Strategies used in the game of *RISK* via Markovian Analysis and Monte-Carlo Simulation

GRADUATE RESEARCH PAPER

Jordan D. Lee, Major, USAF
AFIT/IOA/ENS/12-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/IOA/ENS/12-02

The Comparison of Strategies used in the game of RISK via Markovian Analysis and Monte-Carlo Simulation

GRADUATE RESEARCH PAPER

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Analysis

Jordan D. Lee, BA, MA

Major, USAF

June 2012

AFIT/IOA/ENS/12-02

The Comparison of Strategies used in the game of RISK
via Markovian Analysis and Monte-Carlo Simulation

Jordan D. Lee, BA, MA
Major, USAF

Approved:

//SIGNED//	8 JUNE 2012
Dr. James W. Chrissis (Chairman)	date

Abstract

This paper analyzes strategies of the boardgame *RISK* using Markov chain analysis and Monte-Carlo simulation in order to compare state-based strategies against sequentially dependent or non-memoryless strategy policies. Previous work had focused on calculating the probability of winning based on using all available engagement strategies and battling until either the attacker is unable to continue engaging the enemy or until the defender is annihilated. This research project applied decision analysis methods to look at alternate strategy policies.

Two primary models were utilized to analyze these strategy policies. First, a computer model was developed that would build a Markov chain with the associated transition probabilities based on an initial set of conditions and a specified set of rolling strategies. Second, a Monte-Carlo simulation was developed that would simulate rolling the dice in order to analyze sequentially dependent strategy policies that cannot be modeled via Markov chains. These strategies were then compared based on the attacker's probability of winning and the expected difference between force strengths at the end of a series of engagements.

Contents

List of Figures

List of Tables

The Comparison of Strategies used in the game of RISK via Markovian Analysis and Monte-Carlo Simulation

I. Introduction

Background

The purpose of this project was to analyze the operational strategies of the stochastic wargame *RISK*. Wargaming is typically used to analyze future requirements, assess current capabilities and vulnerabilities, and to train specific tactics, techniques, and procedures based on a "real-world" or hypothetical scenario and/or operations plan. However, wargaming can also be used to teach strategic thinking and the principles of quality decision making.

Historically, games like chess have been used to sharpen one's ability to think strategically and to form logical courses of action. However, chess is only a wargame in a limited sense. When the game begins, both opponents have complete information, and the battle outcomes are deterministic (Herman, et al. 2008)

On the other hand, modern wargames are vastly more realistic and integrate stochastic effects into the design of the game. Computer based wargames typically use complex "black box" algorithms to determine battle outcomes. Therefore, the ideal wargame to study is one where the battle outcomes are stochastically determined, but not too complex to be analyzed and discussed in an academic setting.

For this project, the board game RISK was used to analyze operational strategies in a stochastic wargame environment. The scenario presented in the game of RISK is well-known and does not require any specialized training or skills to play. Furthermore, the stochastic effects (i.e. the "black box" algorithms) can be modeled exactly.

In terms of military doctrine, *RISK* effectively models the three levels of war. First, a player must develop strategies to use at the strategic or global level in order to effectively determine which territories to fortify, which territories to invade, and what types of alliances to form with other players. Although the development of global strategies is outside of the scope of this paper, further information on this topic can be found in (Honary, 2010).

Secondly, a player must develop operational strategies to determine how the conquest of opposing territories can be linked to meet the overall strategic objective of winning the war. Additionally, it is at the operational level that the player must decide which tactical strategies to employ as well as the required force strength to conduct any potential operation. Thirdly, a player must develop tactical strategies to determine how many armies to use during each engagement.

Rules of *RISK*

The game of *RISK* was introduced to the United States in 1959. The game is designed to be played by 2 to 6 players. Each player has color-coded pieces which represent the number of armies each player has. Over the years the game pieces have changed slightly, with different types of pieces or figures representing a different number of armies. One of the most familiar sets of game pieces consists of figures where an infantryman represents one army, a cavalryman represents five armies, and an artillery piece represents ten armies.

The game board has also seen minor changes with regards to theme and layout; however, all of these games share the commonality of 42 territories across six different continents. The game setup has evolved slightly over time. Originally, playing cards that

represented each territory were shuffled and distributed to each player. These cards represented the players' initial set of territories. When the game was revised in 1963, cards were no longer used to determine starting positions; rather each player in turn was able to pick any unoccupied territory until all territories had been claimed.

Additionally, the number of reinforcements a player receives on his or her turn has remained relatively unchanged. A player receives one army for every third territory he or she occupies. A player can also choose to turn in playing cards (which are awarded if a player defeats another player on his or her turn) for a specified number of armies.

However, the way the game is played has remained consistent. According to the official rules of the game, there are seven phases during a player's turn. Those phases are as followed:

1. Determine the number of armies the player is entitled to receive.
2. Place these armies in any of the territories the player occupies.
3. Attack another player's territory.
    a. A player can only attack from an adjacent territory.
    b. At least one army must remain in the currently occupied territory and cannot be used to attack.
4. Cease attacking the other players, either by choice or because the player does not have any enough armies.
5. Make a free move. The player can move any number of armies from one territory to one adjacent territory. Note: at least one army must be left in each occupied territory.

6. If the player defeated another player during his or her turn, then that player receives a playing card. Note: only one playing card can be received during each turn.
7. The player ends his or her turn by passing the "attacker's" dice to the next player. (Parker Brothers, 1959)

Table 1 shows some of the differences in the rules of the game based on the revision year.

	1959	1963	1975	1980	1993	2008
Game Pieces	Cube = 1 Army Oblong = 10 Armies	✈ = 1 Army ★ = 10 Armies		Roman Numerals (I, III, V, and X)	Infantry = 1 Army; Cavalry = 5 Armies Artilerry = 10 Armies	> = 1 Army >>> = 3 Armies
Map	6 Continents / 42 Territories					
Cards	42 terriritory cards (with footsoldier, horseman, or cannor) + 2 Joker Cards (with all three figures)					42 Territory cards
Setup	Divide the cards; place one army per territory on the cards	Players choose	Added "Allied Army" for 2 person play	Introduced game variations	"Secret Mission" *RISK*	
Additional Armies	1 Army / 3 occupied territories; continent bonus; increasing card bonus					City and capital bonuses; different card bonuses
Free Move	Move armies from 1 territory to an adjacent territory					Move armies to a "connected" territory

Table 1 - Summary of Different Rules of *RISK*

Battle Calculus

Before looking at the rules that govern how battles are conducted, two terms need to be clarified. First, the term *engagement* is defined as a single battle between two opposing forces and refers to a single roll of the dice by both players. Second, the term *operation* is defined as a series of battles between two opposing territories. Note that a single operation can consist of several engagements, and a player can also choose to conduct several operations per turn.

Additionally, for the purpose of discussions throughout this paper, the following variables are used to express force strength and attrition:

1. Attacker state variables:
    - N_A' = Total number of attacker's armies
    - N_A = The number of attacker's armies available for a specific operation
    - n_a = The number of attacker's armies used during an engagement
    - The number of attacker's armies lost during a specific operation
    - L_a = The number of attacker's armies lost during an engagement
2. Defender state variables:
    - N_D' = Total number of defender's armies
    - N_D = Total number of defender's armies available for a specific operation
    - n_d = Total number of defender's armies used during an engagement
    - L_D = Total number of defender's armies lost during a specific operation
    - L_d = Total number of defender's armies lost during an engagement

The rules that govern the conduct of battle in the game of RISK are as follows. First, the attacker chooses how many armies will attack and rolls one die for each army in the engagement. The attacker can only engage with a maximum of three armies, even if more armies are available. Additionally, the attacker cannot attack with the last remaining army in that territory. To put these rules in terms of the previously defined variables:

1. $n_a \leq 3$
2. $n_a \leq N_A - 1$

Second, the defender chooses how many armies will defend and rolls one die for each army in the engagement. The defender can use all available armies in the defending territory. To put these rules in terms of the previously defined variables:

1. $n_d \leq 2$
2. $n_d \leq N_D$

The rules that govern battle calculus are as follows: First, each side rank orders their respective dice from highest to lowest. Next, they compare the highest die rolled on each side, and the higher die wins that portion of the engagement. If both sides rolled more than one die, then they compare the next highest die rolled, and the higher die wins that portion of the engagement. If the compared dice are equal, then the defender wins the tie breaker. It is possible for one side to win the first portion of the engagement and the other side to win the second. In that case both sides lose one army each. It is important to note that maximum number of total armies lost is equals the minimum number rolled by either side.

The following figure has three examples of various battle outcomes based on the dice combinations rolled.

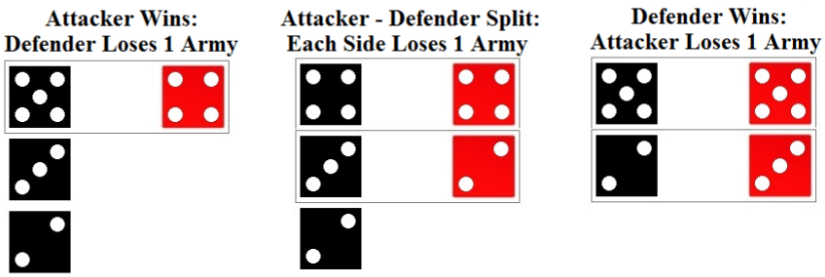

Figure 1 - Battle Outcome Examples

Each battle outcome can be expressed in terms of the associated probability of winning, losing, or tying. Theses probabilities can easily be calculated directly or by enumerating every possible combination.

The possible outcomes and the associated probabilities based on a single attacker fighting against a single defender is displayed if Figure 2. In this case there is a .417 probability that the attacker will win and the defender will lose one army, and a .583 probability that the defender will win and the attacker will lose one army.

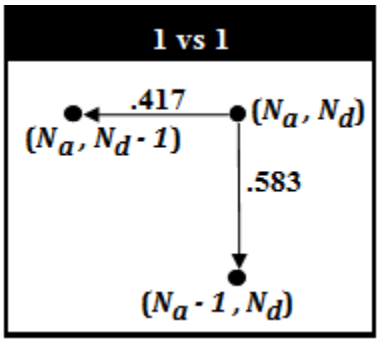

Figure 2 - Probability Outcomes (1 vs 1)

The possible outcomes and the associated probabilities based on two attacking armies fighting against a single defender is displayed if Figure 3. In this case there is a .579 probability that the attacker will win and the defender will lose one army, and a .421 probability that the defender will win and the attacker will lose one army.

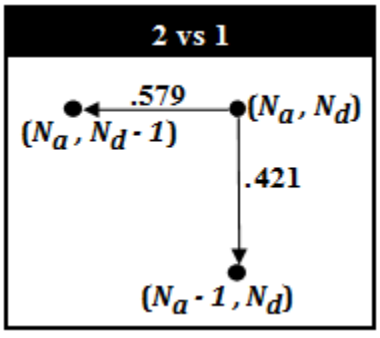

Figure 3 – Probability Outcomes (2 vs 1)

The possible outcomes and the associated probabilities based on three attacking armies fighting against a single defender is displayed if Figure 4. In this case there is a .660 probability that the attacker will win and the defender will lose one army, and a .340 probability that the defender will win and the attacker will lose one army.

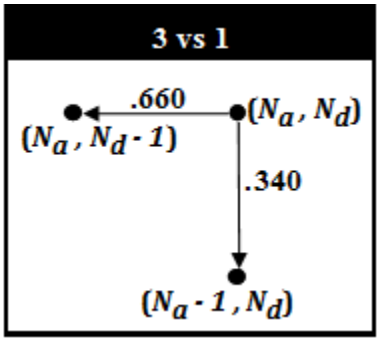

Figure 4 - Probability Outcomes (3 vs 1)

The possible outcomes and the associated probabilities based on a single attacker fighting against a two defending armies is displayed if Figure 5. In this case there is a .225 probability that the attacker will win and the defender will lose one army, and a .745 probability that the defender will win and the attacker will lose one army.

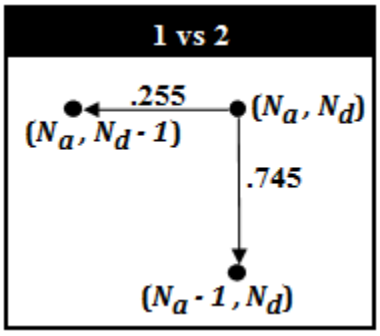

Figure 5 - Probability Outcomes (1 vs 2)

The possible outcomes and the associated probabilities based on a two attacking armies fighting against two defending armies is displayed if Figure 6. In this case there are three possible outcomes. There is a .228 probability that the attacker will win and the defender will lose two armies, a .448 probability that the defender will win and the

attacker will lose two armies, and a .324 probability that both sides will one portion of the engagement and both sides will lose one army.

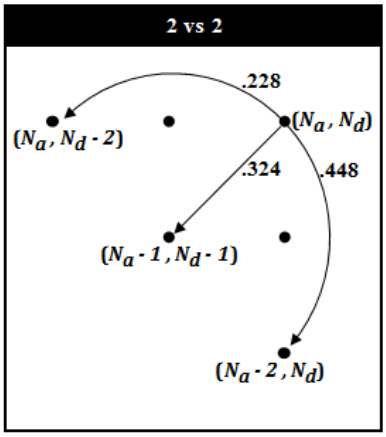

Figure 6 - Probability Outcomes (2 vs 2)

Finally, the possible outcomes and the associated probabilities based on a three attacking armies fighting against two defending armies is displayed if Figure 7. Once again, there are three possible outcomes. There is a .372 probability that the attacker will win and the defender will lose two armies, a .293 probability that the defender will win and the attacker will lose two armies, and a .336 probability that both sides will one portion of the engagement and both sides will lose one army.

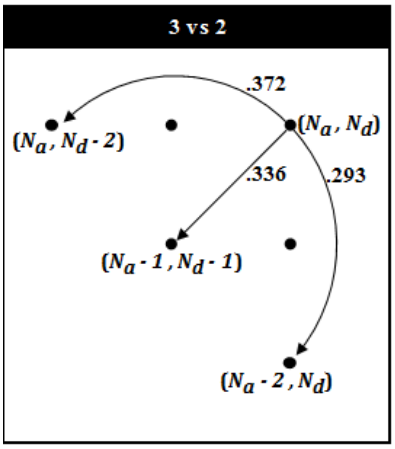

Figure 7 - Probability Outcomes (3 vs 2)

A consolidated list of the battle outcome probabilities are shown in Table 2.

n_a vs n_d	P(A Wins)	P(Split)	P(D Wins)
3 vs 2	2890/7776	2611/7776	2275/7776
3 vs 1	855/1296	-	441/1296
2 vs 2	295/1296	420/1296	581/1296
2 vs 1	125/216	-	91/216
1 vs 2	55/216	-	161/216
1 vs 1	15/36	-	21/36

n_a vs n_d	P(A Wins)	P(Split)	P(D Wins)
3 vs 2	0.3717	0.3358	0.2926
3 vs 1	0.6597	-	0.3403
2 vs 2	0.2276	0.3241	0.4483
2 vs 1	0.5787	-	0.4213
1 vs 2	0.2546	-	0.7454
1 vs 1	0.4167	-	0.5833

Table 2 - Probability Outcomes

Research Objectives

The game of *RISK* has several characteristics that make it ideally suited for an academic discussion regarding military strategy, military art, decision analysis, and mathematical modeling. The overall objective for this research project is to develop a method for comparing state-based strategies to sequentially dependent policies in a stochastic wargame environment that could be used in an academic or seminar setting to teach principles of operations analysis.

II. Literature Review

Articles

There have been several articles written about calculating the probability of various stochastic outcomes within the context of board games. Ash and Bishop (1972) were the first ones to use Markov chains to find the steady state probabilities of occupying a specific property in the game of Monopoly. For the most part, this was a straight forward computation, with the only assumption pertaining to how long a player stays in jail. Although this was an interesting approach to calculating probability distributions, there are significant differences between the game of Monopoly and the game of RISK that need to be addressed before the Markovian method could be applied to the game RISK. First, in Monopoly each player has to roll the dice. In RISK, each player has a choice whether or not to attack and, if so, how many dice to use. Additionally, RISK should be modeled using absorption probabilities as opposed to steady-state probabilities.

Tan (1997) partially answered these differences by showing how a Markov chain could be modeled using the probabilities associated with the stochastic outcomes of the dice rolls in the game of RISK. Unfortunately, Tan miscalculated the transition probabilities associated with the joint distributions when both players used more than one die during the engagement. Tan's calculations for these joint distributions mistakenly assumed an independent relationship between the highest dice combinations and the second highest dice combinations.

Osborne (2003) corrected Tan's mistake and concluded the same transition probabilities that were calculated in the previous chapter. Osborne calculated the

probability of winning given the initial force strengths of both the attacker and defender; however, Osborne only looked at one strategy policy: the attacker will continue to attack until either the defender has been destroyed or the attacker has no more available armies. It is important to note that Osborne's strategy policy also used one attacking army to fight against two defending armies when the attacker only had one remaining available army and the defender had more than one army. However, Osborne effectively demonstrated that the probabilities associated with the game of RISK can be modeled using a Markov chain.

Blatt (2002) also used Markov chains to calculate the probabilities of winning in the game of RISK. However, Blatt expanded the probability distributions of the associated outcomes by looking at the possibility of using dice that had more than just six sides. The results of this study are listed in the following table:

6 sided dice					Transition	General Formula
Case	a	d	From state	To state	Probability	(s = # of faces on die)
I	2	1	(2,1)	(2,0)	0.4166	$\frac{s-1}{2s}$
				(0,1)	0.5834	$\frac{s+1}{2s}$
II	3	1	(3,1)	(3,0)	0.5787	$\frac{(s-1)(4s+1)}{6s^2}$
				(2,1)	0.4213	$\frac{(s+1)(2s+1)}{6s^2}$
III	≥ 4	1	(a,1)	(a,0)	0.6597	$\frac{(s-1)(3s+1)}{4s^2}$
				(a-1,1)	0.3403	$\frac{(s+1)^2}{4s^2}$
IV	2	≥ 2	(2,d)	(2,d-1)	0.2546	$\frac{(s-1)(2s-1)}{6s^2}$
				(1,d)	0.7454	$\frac{(s+1)(4s-1)}{6s^2}$
V	3	≥ 2	(3,d)	(3,d-2)	0.2276	$\frac{(s-1)(2s^2-2s-1)}{6s^3}$
				(1,d)	0.4483	$\frac{(s+1)(2s^2+2s-1)}{6s^3}$
				(2,d-1)	0.3241	$\frac{(s-1)(s+1)}{3s^2}$
VI	≥ 4	≥ 2	(a,d)	(a,d-2)	0.3717	$\frac{(s-1)(6s^3-3s^2-5s-2)}{12s^4}$
				(a-2,d)	0.2926	$\frac{(s+1)(2s+1)(3s^2+3s-1)}{30s^4}$
				(a-1,d-1)	0.3357	$\frac{(s+1)(s-1)(18s^2+15s+8)}{60s^4}$

Table 3 - General Formula for Computing Transition Probabilities (Blatt, 2002)

Instead of using Markovian analysis, other authors utilized dynamic programming to model RISK. Interestingly, the rules associated with RISK are not the same around the world. Koole (1994) discusses an optimal dice rolling policy for the Dutch version. Dutch rules allow the defender to roll the second die after seeing the dice roll outcomes of the attacker. Consequently, the defender would only want to roll the second die if the second highest attacker die was less than four, which would favor the defender's chance of success.

Malliphant and Smith (1990) also used dynamic programming to model RISK. Their analysis included the probability of success for an attacker playing optimally and stopping when the attacker's strength dropped below that of the defender. The results from this analysis will compared to other strategies later in this paper.

Markov Chains Theory

According to Kemeny and Snell (1976), a Markov process is memoryless such that $[f_{n+1} = s_{j+1} \mid (f_n = s_i)\ ,\ (f_{n-1} = s_{i-1}),\ (f_{n-2} = s_{i-2})\ ,\ldots,\ (f_o = s_o)\ ] = P[f_{n+1} = s_{j+1} \mid (f_n = s_i)]$. An absorbing Markov chain can be represented by the transition matrix

$$P = \begin{bmatrix} Q & R \\ O & I \end{bmatrix}$$

where Q is the matrix of transition probabilities among the transient states, R is the matrix of transition probabilities for absorption from the transient states, 0 is a matrix of zeroes, and I is an identity matrix (representing transitions within the absorbing states). Partitioning P in this manner allows for relatively easy computation of key operating characteristics of the Markov chain. Specifically, the absorption probabilities can be computed in the following manner.

Let a_{ij} =P[process enters absorbing state i given that the initial state is j]. These probabilities can be expressed by the system of equations:

$$a_{ij} = p_{ij} + \sum_{k} p_{ik} * a_{kj}$$

which can be solved recursively (k is indexed over all transient states). This system can be written in matrix form as:

$$A = R + QA$$

Solving for A gives

$$A = [I - Q]^{-1}R$$

and the corresponding absorption probabilities of interest can be picked out of the matrix .

III. Methodology

Markov Chain Method

Because one of the purposes of this paper is to compare state-based strategies with sequentially-dependent policies, the Markovian analysis summarized during the literature review was determined to be the best method for modeling the state-based strategies. The states of the Markov process that models an engagement corresponds to the number of armies each player has available for use in that particular engagement. A transition event corresponds to the roll of the dice within a particular engagement. Because the engagement is always initiated and broken off by the attacker, the point at which the attacker ceases the attack is considered an absorbing state. These absorption states correspond either to a state where the attacker no longer seeks conquest over the opponent's territory or when the defender is defeated.

Consistent with the rules of the game, an attacker can never have fewer than two armies (the attacker cannot attack using the last remaining army since at least one army must be left behind). Thus we have a two-dimensional state space N = {number of armies available to the attacker; number of armies available to the defender} = {N_A vs N_D}. Using these descriptions, the set of states for the attacker is N(A) = {2, 3, • • •,M} and for the defender N(D) = {0, 1, 2, • • •,N}. Thus, the two-dimensional state space is described by the set of 2-tuples, X_k = {(m, n) | m ∈ N(A), n ∈ N(D)} on the kth transition. The transition probabilities are P[N_A = r, N_D = s) | N_A = m, N_D = n)] which are represented by P[(r, s), (m, n)]. The transition probabilities are computed based on the attack and defend strategies to be employed during the play of the game.

The set of states to which transitions can occur and the probabilities of those transitions are dependent on the specific attacker and defender strategies that are employed. It should also be evident that since these attrition probabilities are based on pure chance (the dice rolls), the transition probabilities are stationary.

Given the attrition probabilities stated in the previous sections, the attacker and defender strategies, and the cutoff rule(s) to be employed, various performance characteristics for the system can be calculated. These include the probability of the attacker winning or losing a specific operation, as well as other expected end state strength and expected losses. All of these calculations can be derived through Markov chain analysis. The basic Markov chain analysis is presented and specific examples follow.

Classifying the states of the Markov process, we have that all states corresponding to engagements are transient states, with absorbing states for those states where the engagement terminates, either for the attacker breaking off the attack, or the defender being annihilated. As previously stated, the attrition is a “must die” circumstance, so that transitions during any particular engagement can never increase armies for either side, and the system must ultimately land in one or the other of the absorbing classes, where either the attacker wins or loses the battle. The absorbing class corresponding to the attacker win situation always occurs when, through a series of engagements the defender’s force is ultimately wiped out, and the attacker occupies that territory.

A. These probabilities are used extensively in subsequent analyses in this paper.

It is then relatively simple to create a matrix of the end states associated with the absorption matrix.

$$S = \begin{bmatrix} N_{A_{s1}} & N_{A_{s2}} & \dots & N_{A_{sn}} \\ N_{D_{s1}} & N_{D_{s2}} & \dots & N_{D_{sn}} \end{bmatrix}$$

where *S* represents the matrix of end states, each element in the top row is the attacker's end state that is associated with the absorption matrix, and each element in the bottom the defender's end state that is associated with the absorption matrix. When the absorption and end state matrices are multiplied, the product is the expected end state for both the attacker and defender:

$$E[S] = A * S^T$$

Consequently, by subtracting this number from the initial force strength, the result is the expected losses for both sides.

$$E[L_A] = N_{A_0} - E[S_A]$$

$$E[L_D] = N_{D_0} - E[S_D]$$

Markov Chain Calculations

The Markov chain then becomes an operation plan that outlines what tactical strategy to utilize given a particular state. Figure 8 shows an operational strategy that utilizes every possible tactical strategy. This mirrors what the previous journal articles discussed.

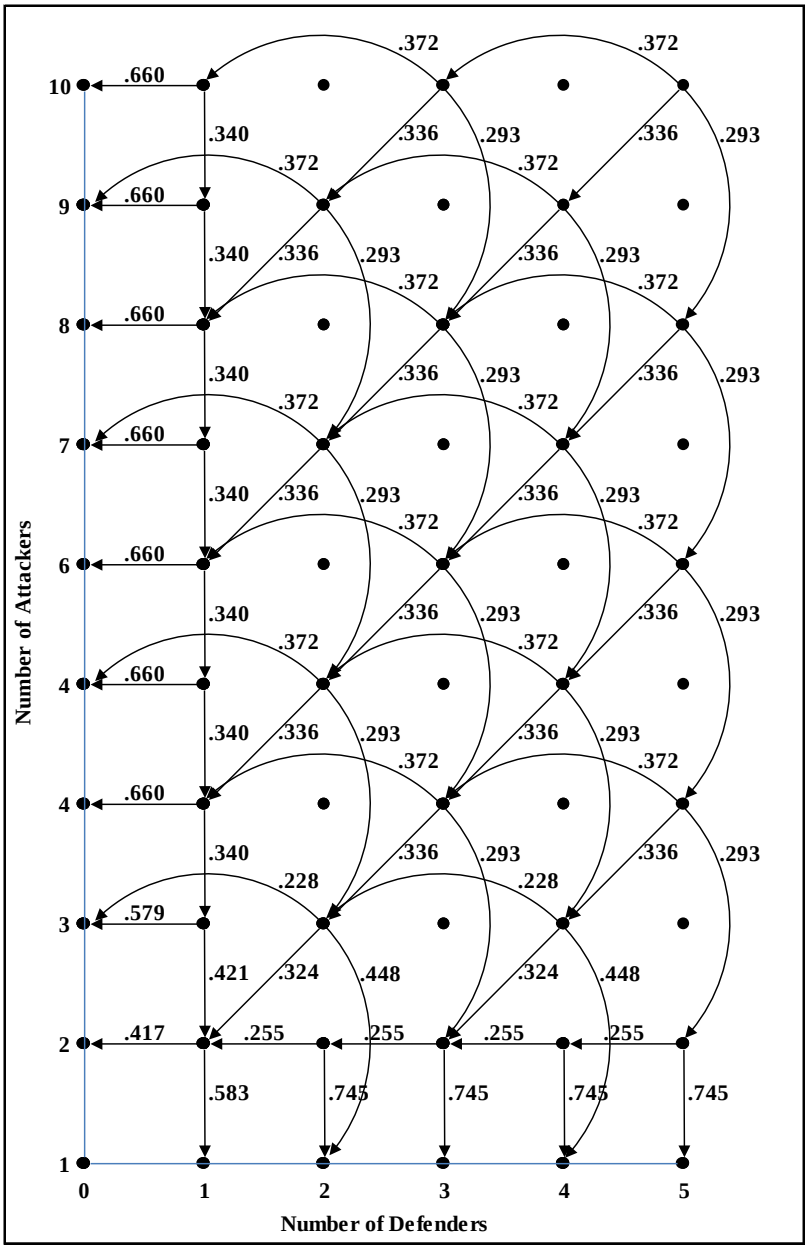

Figure 8 - Markov Chain (utilizing all tactical strategies)

Figure 9 shows an operational strategy the employs all tactical strategies except for a single attacker against two defenders scenario.

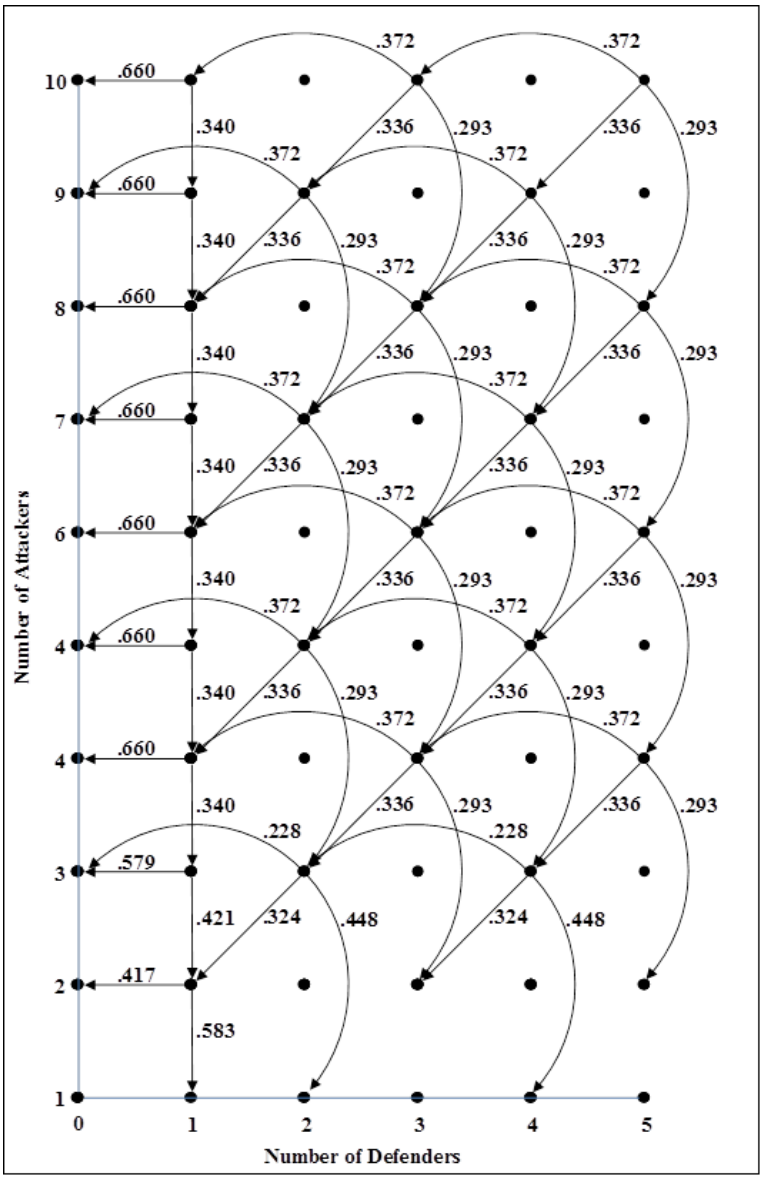

Figure 9 - Markov Chain (excluding 1 vs 2)

The difference between Figure 9 and Figure 8 is the missing row of possible transitions along the bottom row. Figure 10 shows an operational strategy where the attacker ceases attacks when that attacker force strength drops below the defender's force strength.

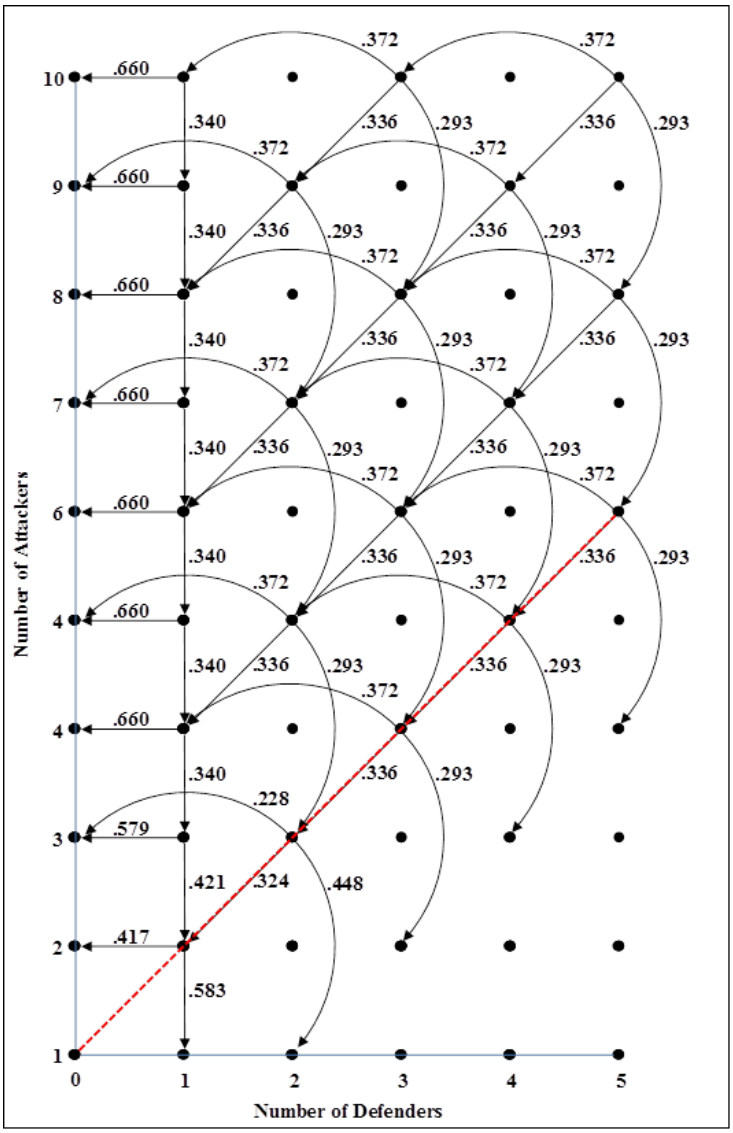

Figure 10 - Markov Chain ($N_A \geq N_D$)

A computer model was developed that would quickly create Markov matrices based on specified initial conditions and selected strategies. The inputs for the model are as follows:

1. Initial attacker force strength: the number of armies the attacking force has initially.

2. Initial defender force strength: the number of armies the defending force has initially.
3. Attacker strategy: the tactical strategies the attacker will employ based on the scenario. This strategy is coded using a six digit string. The first digit represents how many armies the attacker will utilize in a three against one scenario. The second digit represents how many armies the attacker will utilize in a three against two scenario. The third digit represents how many armies the attacker will utilize in a two against one scenario. The fourth digit represents how many armies the attacker will utilize in a two against two scenario. The fifth digit represent how many armies the attacker will utilize in a one against one scenario. Finally, the sixth digit represents how many armies the attacker will utilize in a one against two scenario. For example, the first Markov chain created in Figure 8 would be coded as 332211; whereas the Markov chain created in Figure 9 would be coded as 332210.
4. Defender strategy: the tactical strategy the defender will employ. The defender only has two options. The defender can use two armies until only one is available, or the defender can always choose to use only one army.
5. Independent versus dependent strategy: determines if the attacker's strategy should be modeled independent or dependent upon the defender's strategy. It is only a factor when the defender chooses to defend with only one army.
6. Attacker end state: the minimum number of required attacking armies, which could be based on a constant value, based on a percentage of defender strength, or a combination of both.

7. Initial attacker attrition: creates an absorbing state if the attacker chooses to cease engagements if the attacker force drops below a specified value and the defender has not lost any engagements.

A picture of the graphical user interface used by the program is displayed in Figure 11.

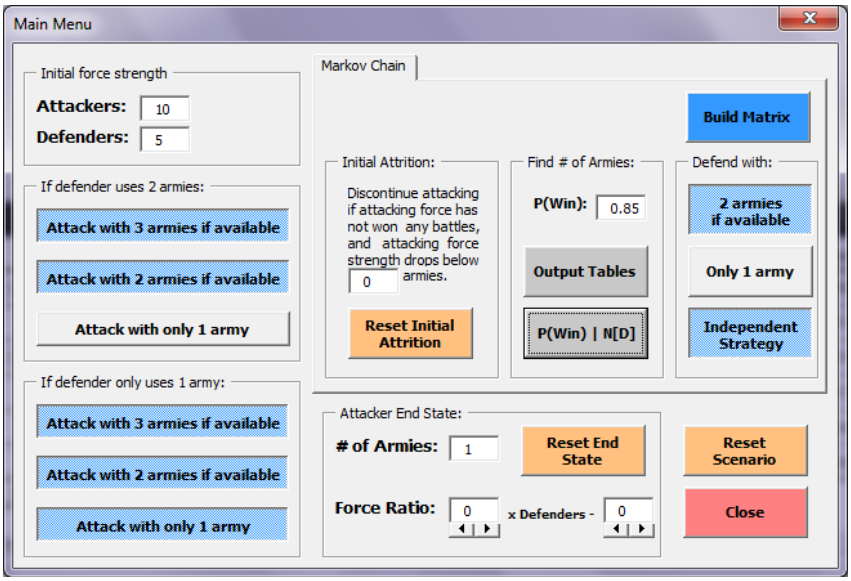

Figure 11 - Markov Model Interface

First, the Markov program will construct a P-matrix and fill in the appropriate transition probabilities. An example based on a $(N_A = 5, N_D = 2)$ initial state is shown in Figure 12.

P-Matrix	5 vs 2	4 vs 1	3 vs 2	3 vs 1	2 vs 1	1 vs 2	1 vs 1	5 vs 0	4 vs 0	3 vs 0	2 vs 0
5 vs 2	0	0.3357767	0.2925669	0	0	0	0	0.3716564	0	0	0
4 vs 1	0	0	0	0.3402778	0	0	0	0	0.6597222	0	0
3 vs 2	0	0	0	0	0.3240741	0.4483025	0	0	0	0.2276235	0
3 vs 1	0	0	0	0	0.4212963	0	0	0	0	0.5787037	0
2 vs 1	0	0	0	0	0	0	0.5833333	0	0	0	0.4166667
1 vs 2	0	0	0	0	0	1	0	0	0	0	0
1 vs 1	0	0	0	0	0	0	1	0	0	0	0
5 vs 0	0	0	0	0	0	0	0	1	0	0	0
4 vs 0	0	0	0	0	0	0	0	0	1	0	0
3 vs 0	0	0	0	0	0	0	0	0	0	1	0
2 vs 0	0	0	0	0	0	0	0	0	0	0	1

Figure 12 - P-Matrix Output

The program then splits the P-matrix into the Q-, R-, 0-, and I- sub-matrices as shown in Figure 13.

Q-Matrix	5 vs 2	4 vs 1	3 vs 2	3 vs 1	2 vs 1
5 vs 2	0	0.3357767	0.2925669	0	0
4 vs 1	0	0	0	0.3402778	0
3 vs 2	0	0	0	0	0.3240741
3 vs 1	0	0	0	0	0.4212963
2 vs 1	0	0	0	0	0

R-Matrix	1 vs 2	1 vs 1	5 vs 0	4 vs 0	3 vs 0	2 vs 0
5 vs 2	0	0	0.3716564	0	0	0
4 vs 1	0	0	0	0.6597222	0	0
3 vs 2	0.4483025	0	0	0	0.2276235	0
3 vs 1	0	0	0	0	0.5787037	0
2 vs 1	0	0.5833333	0	0	0	0.4166667

0-Matrix	5 vs 2	4 vs 1	3 vs 2	3 vs 1	2 vs 1
1 vs 2	0	0	0	0	0
1 vs 1	0	0	0	0	0
5 vs 0	0	0	0	0	0
4 vs 0	0	0	0	0	0
3 vs 0	0	0	0	0	0
2 vs 0	0	0	0	0	0

I-Matrix	1 vs 2	1 vs 1	5 vs 0	4 vs 0	3 vs 0	2 vs 0
1 vs 2	1	0	0	0	0	0
1 vs 1	0	1	0	0	0	0
5 vs 0	0	0	1	0	0	0
4 vs 0	0	0	0	1	0	0
3 vs 0	0	0	0	0	1	0
2 vs 0	0	0	0	0	0	1

Figure 13 - P-Matrix Sub-Matrices

The model then calculates the fundamental (N-matrix) and absorption (A-matrix) matrices as shown in Figures 14 and 15.

N-Matrix	5 vs 2	4 vs 1	3 vs 2	3 vs 1	2 vs 1
5 vs 2	1	0.3357767	0.2925669	0.1142574	0.1429495
4 vs 1	0	1	0	0.3402778	0.1433578
3 vs 2	0	0	1	0	0.3240741
3 vs 1	0	0	0	1	0.4212963
2 vs 1	0	0	0	0	1

Figure 14 - N-Matrix Output

A-Matrix	1 vs 2	1 vs 1	5 vs 0	4 vs 0	3 vs 0	2 vs 0
5 vs 2	0.1311585	0.0833872	0.3716564	0.2215194	0.1327162	0.0595623
4 vs 1	0	0.0836254	0	0.6597222	0.19692	0.0597324
3 vs 2	0.4483025	0.1890432	0	0	0.2276235	0.1350309
3 vs 1	0	0.2457562	0	0	0.5787037	0.1755401
2 vs 1	0	0.5833333	0	0	0	0.4166667

Figure 15 - A-Matrix Output

Now that the Markov chain model had been created, it was time to construct and examine some baseline state-based strategic policies.

Strategy Development

Maliphant and Smith (1990) discussed the following four possible objectives which could affect a player's strategy:

1. *Maximize the probability that the attacker defeats the defender;*
2. *Maximize the expected number of pieces in the attacker's army at the end of the turn;*
3. *Maximize the expected difference between the two armies at the end of the turn;*
4. *Minimize the number of expected number of pieces in the defender's army at the end of the turn.*

With the exception of Maliphant and Smith, the other articles discussed previously were only concerned with strategies that maximized the probability of the attacker defeating the defender; however, this research project also focused on strategies based on maximizing the expected difference between the two armies at the end of the turn. For the purpose of further discussion, the following variables need to be defined:

1. Δ'_A= Difference of total force strength (global): $(N'_A - N'_D)$
2. Δ_A= Difference of concentrated force strength (operational): $(N_A - N_D)$
3. δ_A = The actually amount force delta variables are increased based on a specific operational outcome: $\delta_A = \left(L_D - L_A\right)$
4. $E[\delta_A]$ = The expected amount force delta variables will be increased based on a specific operational strategy: $E[\delta_A] = \mathrm{E}\left[L_D - L_A\right]$

5. ε = The expected amount force delta variables will be increase based on a specific tactical strategy: $\varepsilon = \mathrm{E}[L_d - L_a]$

These newly defined force delta variables are from the attacker's perspective as opposed to an absolute value. A positive delta indicates that the attacker's force strength is superior to the defender's force strength. It is possible that even though the total force delta may be positive, that individual operational deltas may be negative indicating that the attacker would be outnumbered in that particular operation.

The overall objective of the attacker can be summarized by $N'_A = \Delta'_A$. Likewise, the operational objective of the attacker can be defined as $N_A = \Delta_A$. In order to accomplish this, the attacker's optimal strategy would be to maximize both $E[\delta_A]$ and ε. Another way to view $E[\delta_A]$ would be in terms of a utility function (see Figure 16) where the upper branch is the $E[\delta_A]$ given the attacker wins and the lower branch is $E[\delta_A]$ given the defender wins. Although this project only refers to this function based on a risk neutral risk attitude, future studies could expand this model by looking at other risk attitudes.

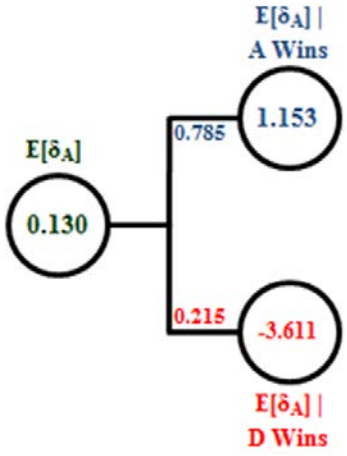

Figure 16 - Utility Function

Game Theory

An interesting way to analyze tactical strategies is to use game theory. Von Neumann and Morgenstern (1972) developed a theory for formulating optimal strategies during a two-person zero-sum game. The theory looks at a two-person zero-game from the perspective of two players, a row player and a column player. The expected reward for the row player is equal to the loss of the column player. The theory proposes that each player should choose a strategy that limits potential loss rather than maximize potential gain. In the example from their book (see Table 4), the row player would find the minimum payoff from each row and then choose the row that maximized the minimum value.

R \ C	1	2	row minima
1	-2	2	-2
2	-1	2	-1
column maxima	-1	2	

Table 4 - Game Theory Example

In this case, the row player would want to choose the second row, because its minimum value is higher than the minimum value in the first row. Conversely, the column player would find the maximum loss from each column and then choose the column that minimized the maximum loss value. In this case, the column player would want to choose the first column, because its maximum value is lower than the maximum value in the second column. This example also shows that these players would want to

choose the same strategy every time, creating a saddle point where the expected value of the game is the same for both players.

For the purposes of *RISK*, by calculated the expected losses for both sides (see Tables 5 and 6) based on a particular tactical strategy, an expected difference in attrition rates for both sides can be compared.

n_a vs n_d	$E[L_a]$ | A Wins	$E[L_a]$ | Split	$E[L_a]$ | D Wins	$E[L_a]$
3 vs 2	0	0.3358	0.5851	0.9209
3 vs 1	0	-	0.3403	0.3403
2 vs 2	0	0.3241	0.8966	1.2207
2 vs 1	0	-	0.4213	0.4213
1 vs 2	0	-	0.7454	0.7454
1 vs 1	0	-	0.5833	0.5833

Table 5 - Expected Attacker Losses

n_a vs n_d	$E[L_d]$ | A Wins	$E[L_d]$ | Split	$E[L_d]$ | D Wins	$E[L_d]$
3 vs 2	0.7433	0.3358	0	1.0791
3 vs 1	0.6597	-	0	0.6597
2 vs 2	0.4552	0.3241	0	0.7793
2 vs 1	0.5787	-	0	0.5787
1 vs 2	0.2546	-	0	0.2546
1 vs 1	0.4167	-	0	0.4167

Table 6 - Expected Defender Losses

These attrition rates can be viewed in terms of a baseline equal attrition rate for both sides with an additional specified reward. That reward would be added to the defender's baseline attrition rate. Conversely, the reward would be subtracted from the attacker's baseline attrition rate.

$$E[L_a] = \frac{1}{2}(L_t - \varepsilon)$$

$$E[L_d] = \frac{1}{2}(L_t + \varepsilon)$$

where L_a = Loss of Attacker's armies
L_d = Loss of Defender's armies
L_t = Total losses = $L_A + L_D$
ε = Reward of tactic/strategy

For a positive reward value, the expected attrition rate for the defender would be greater than the expected attrition rate for the attacker. Solving the above equations for epsilon reveals the expected reward equation:

$$\varepsilon = E[L_d] - E[L_a]$$

Table 7 lists the expected reward values for each tactical strategy.

n_a vs n_d	ε
3 vs 2	0.1582
3 vs 1	0.3194
2 vs 2	-0.4414
2 vs 1	0.1574
1 vs 2	-0.4907
1 vs 1	-0.1667

Table 7 - Expected Reward

In terms of game theory, the attacker would assume the role of the row player and the defender would assume the role of the column player. Putting the expected reward values into the game matrix reveals an optimal strategy, and consequently a saddle point, where the attacker should always attack with three armies and the defender should always defend with two armies (see Table 8).

$n_a \setminus n_d$	1	2	row minima
0	0	0	0
1	-0.1667	-0.4907	-0.4907
2	0.1574	-0.4414	-0.4414
3	0.3194	0.1582	0.1582
column maxima	0.3194	0.1582	

Table 8 - RISK Game Theory Matrix

By adding the possibility of not attacking, if the attacker does not have the ability to attack with three armies, game theory would suggest the optimal strategy is to choose not to attack (see Table 9).

$n_a \setminus n_d$	1	2	row minima
0	0	0	0
1	-0.1667	-0.4907	-0.4907
2	0.1574	-0.4414	-0.4414
column maxima	0.1574	0	

Table 9 - RISK Game Theory Matrix (w/o 3 Attackers)

IV. Analysis

Markovian Analysis

The Markov model was designed to build probability and expected value tables ranging from 2 – 60 attackers and 1 to 30 defenders. The model was then run under the following configurations:

1. Tactical strategies employed: 332211, 332210, 332010, 332200, 332000, 330000.
2. Attacker end state ranging from 1 – 4 remaining armies.
3. Force ratio ranging from 0%- 150% in 25% increments with a constant decrement of minus one army. With a constant decrement of minus one, the attacker would cease engagements if force strength dropped below the specified ratio (allowing 1 attacker vs 2 defenders) or at or below the specified ratio (not allowing 1 attacker vs 2 defenders).

A comparison of all 168 configurations was conducted. For every initial state, strategy 332211 always yielded the highest probability of winning. For every state when the initial attacker strength was greater than four, or when initial attacker strength was three and initial defender strength was one, the highest delta was using strategy 332000. In all other situations, the strategies that produced the highest delta resulted in a delta less than zero, meaning the attacker was expected to lose more armies during the operation than the defender.

The probability that the attacker wins was then plotted against the expected losses delta. The following graph shows the two extreme points. Because no other strategy produced a higher probability of winning than strategy 332211, and no other strategy produced a higher expected losses delta than strategy 332000, a line was drawn between

those two points forming a lower boundary (see Figure 17). Based on the assumption that the player is attempting to maximize his or her probability of winning and/or the expected losses delta, any strategy that falls below this line would be suboptimal and should not be considered.

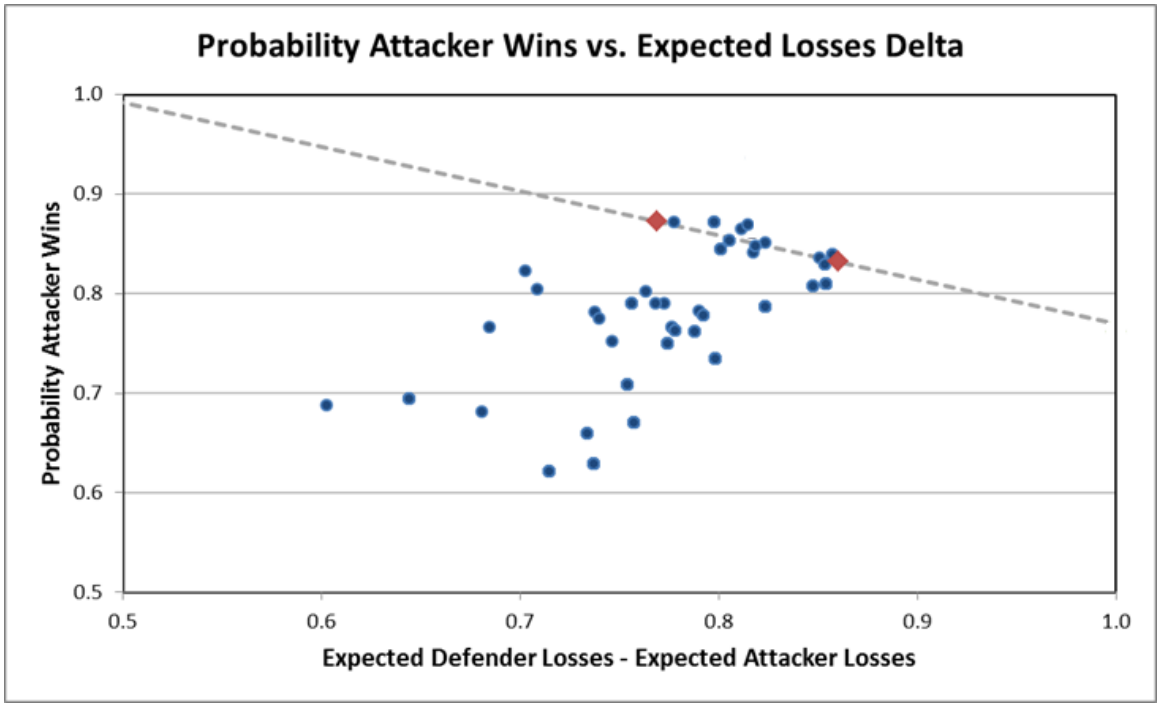

Figure 17 - Strategy Plot (10 vs 5)

One method for determining a strategy that balances these two objectives is to find the point above the boundary line that is furthest from the boundary. To accomplish this calculation, an equation based on a LaGrange multiplier was utilized. According to this method, the minimum or maximum values of a function $F(x,y) = f(x,y) + \lambda\phi(x,y)$, where λ is the LaGrange multiplier and $\phi(x,y)$ equals a constant, can be solved by setting the partial derivatives of F equal to zero. For convenience, instead of calculating the distance from the point to the actual boundary line, an equally appropriate method would be to find the distance from the point to a line parallel to the boundary going

through the origin. Therefore, $f(x,y)$ can be defined as the minimum square distance between the strategy point and the line parallel to the boundary line:

$$f(x,y) = (x_s - x)^2 + (y_s - y)^2$$

where x_s equals the x-coordinate of the strategy point, y_s equals the y-coordinate of the strategy point, x equals the x-coordinate of closest point on the parallel boundary line, and y equals the y-coordinate of the closest point on the parallel boundary line. Additionally, $\phi(x,y)$is defined using the equation of the parallel line:

$$\phi(x,y) = y - m * x$$

where m equals the slope of line between strategy point 332211 and strategy point 332000. The partial derivatives of $F(x,y)$ are as follows:

$$\frac{\partial F}{\partial x} = 2 * (x - x_2) - \lambda * m = 0$$

$$\frac{\partial F}{\partial y} = 2 * (y - y_2) + \lambda = 0$$

Solving these equations and simplifying yields the equation:

$$d = \frac{Y_s - X_s * m}{\sqrt{m^2 + 1}}$$

The strategy points for all 168 combinations were then compared to find the points that were furthest from the boundary line. Those results are listed in Appendix I.

Sequentially Dependent Strategy Analysis

Once the baseline Markovian model had been run and a method for comparing strategies was implemented, the next step was to examine sequentially dependent strategies. Two methods were used. The first, for small initial force strengths, a probability tree could be expanded from the Markov chain into all of its possible branches. Second, for large initial force strengths, a Monte-Carlo simulation was created.

The probability tree method provides exact solutions, but the tree size can quickly get too large for the computer to manage, and the elemental probabilities become so small, round off errors are bound to happen. The Monte Carlo simulation provides an approximate solution with a 95% confidence interval on all batched output.

For example, Figure 18 shows an expanded probability built from a Markov chain with initial force strengths of $N_A = 6$ and $N_D = 2$.

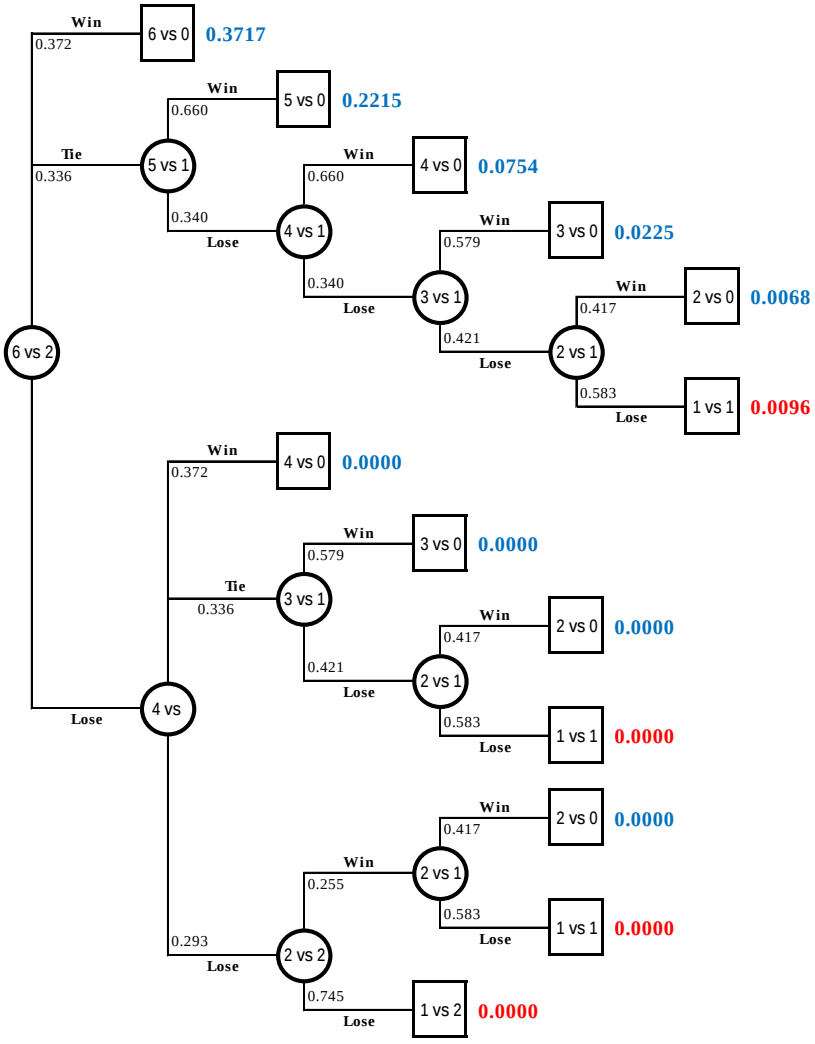

Figure 18 - Markov-Chain Probability Tree

Figure 19 is the same tree with the additional cutoff criteria of the attacker ceasing engagements if he or she loses three or more armies within the first two engagements.

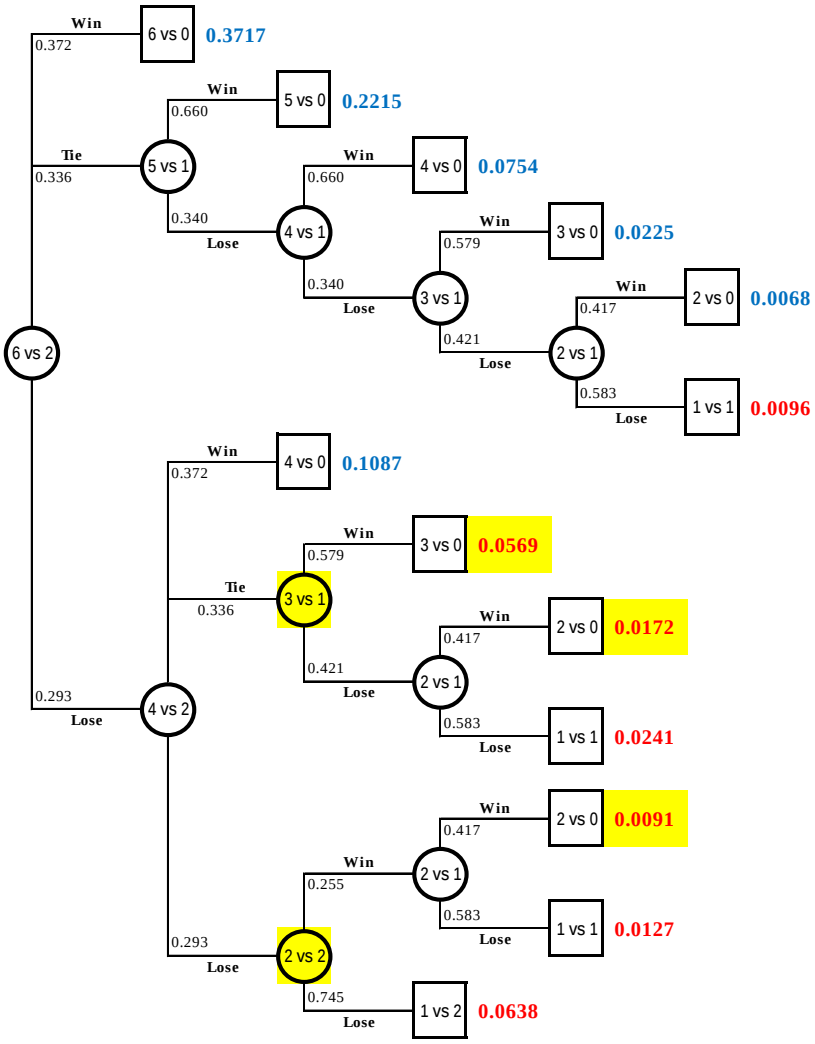

Figure 19 - Sequentially-Dependent Probability Tree

One of the interesting points in this scenario is that the attacker chooses to cease engagements after two rolls due to initial losses at the $N_A = 3$, $N_D = 1$ state. In this case the attacker would still have a slight advantage over the defender. Furthermore, the attacker has decided to stop at $N_A = 3$, $N_D = 1$ in this case, but would choose to fight at that state under a different path.

The computer model essentially builds a truncated P-matrix by building the upper sub-matrices (the Q-matrix and the R-matrix) without all of the zero entries. It first sends the elements from the initial state to their subsequent state. Each time the element enters a new state, it is split or branched to follow the possible paths leading from that state. Each entity keeps track of its current elemental probability, the path it has taken, and its next event. Because the P-matrix created by this model is upper triangular, there is no possibility that an entity could back track to a previous state. Therefore, this method of sequentially sending an entity to its next event can be accomplished without looping the cycle. When the entity reaches an absorbing state, either win or lose, the elemental probability and path taken are stored.

The path taken provides the necessary information to determine attacker and defender end states and losses as well the number of engagements the battle took. An uppercase "W" represents an attacker win where the defender lost two armies, and a lower case "w" represents an attacker win where the defender only lost one army. Likewise, an uppercase "L" represents an attacker loss of two armies, and a lowercase "l'" represents an attacker loss of only one army.

The probability tree was also used to validate the Monte-Carlo simulation. A comparison of calculations is displayed in Table 20.

Monte Carlo		CI(95%)
P[A Wins]:	0.8020	0.0199
P[D Wins]:	0.1980	0.0199

Probability Tree	
P[A Wins]:	0.8066
P[D Wins]:	0.1934

E[X]	μ	CI(95%)
$E[N_A]$	4.588	0.075
$E[L_A]$	1.412	
$E[N_D]$	0.272	0.033
$E[L_D]$	1.728	
$E[\delta_A]$	0.316	0.098

E[X]	μ
$E[N_A]$	4.631
$E[L_A]$	1.369
$E[N_D]$	0.279
$E[L_D]$	1.721
$E[\delta_A]$	0.352

Figure 20 - Probability Tree and Monte-Carlo Comparison

The Monte-Carlo simulation generates dice rolls based on a random number generator designed by L'ecuyer (1998). It was originally programmed in C, but was translated into Visual Basic for Applications to work in this model. This generator was tested using the KS test for uniformity by generating one million random numbers in batches of 100. Each batch was then rank ordered and the maximum difference between the generated continuous probability function and a uniform distribution was calculated. If the maximum difference calculated is less than or equal to than the critical value, then it the test would fail to reject the null hypothesis that the sample was from a uniform distribution (Banks, et al. 2010). The critical value for $\alpha = 0.05$ is 0.136 (for $n = 100$). Out of the 10,000 batches of 100 random numbers, the calculated value was greater than 0.136 only 420 times. This means that the generator exceeded the critical value 4.2% of the time, which makes sense because the critical value was based on the sample not exceeding that value more than 5% of the time.

The random number generator was also tested by simulating the dice roll for a specific tactical strategy one million times per strategy. All of the confidence intervals from those runs included the actual known probabilities of the specific outcomes.

Figure 21 shows the basic flow of the simulation processing.

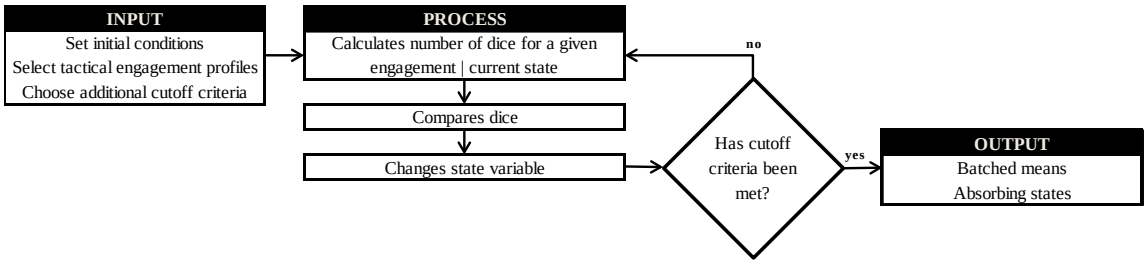

Figure 21 - Monte-Carlo Simulation Diagram

This simulation was validated by comparing the output from several different runs without using any additional cutoff criteria against the Markov model. For example, the simulation was run with an initial condition of N_A = 10 and N_D = 10 and compared to the Markov model. Those results are listed in Table 10. The Monte Carlo simulation output was batched and it calculated a 95% confidence interval.

	Markov Chain	Monte Carlo
P[Attacker Wins]:	0.4685	0.4669 +/- 0.0030
E[Attacker End State]:	3.35	3.3388 +/- 0.0138
E[Defender End State]:	2.39	2.3829 +/- 0.0168
E[Troop Losses]:	6.65	6.6612 +/- 0.0138
E[Number of Battles]:	7.38	7.3862 +/- 0.0079

Table 10 - Monte Carlo Validation

Case Studies

The simulation was then used to analyze four primary cases.

1. Case 1

		Attacker	Defender
Case 1	Start with:	10	10
	Fight with:	Maximum available (will not allow 1 vs 2)	Randomly selecting one or two armies to defend
	Win criteria:	Defender annihilation	Attacker breaks off

The simulation was run several times, each time incrementally increasing the probability that the defender would choose to defend with two armies. The results are graphed in figure 22. As was expected, the defender's odds of winning battles greatly increases as the probability of choosing two armies to defend with approaches 100%.

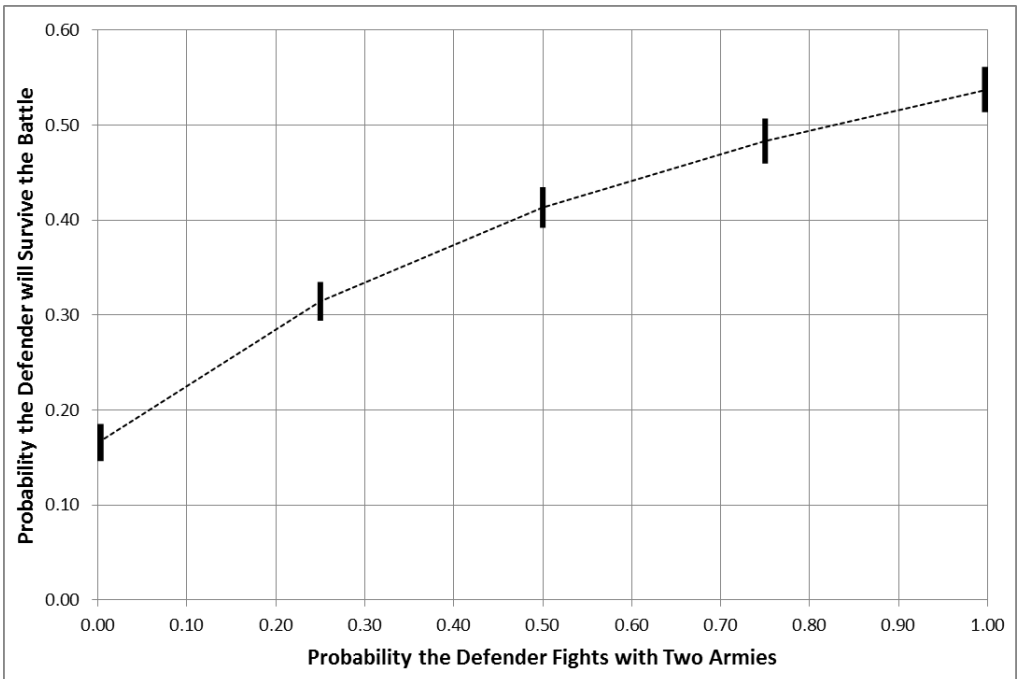

Figure 22 - Probability of Defender Using Two Armies

2. Case 2

		Attacker	Defender
Case 2	Start with:	10	5
	Fight with:	Maximum available (will not allow 1 vs 2)	Maximum Available
	Fight until:	3 total losses	
	Win criteria:	Defender annihilation	Attacker breaks off

Typically when players decide to use additional cutoff criteria, it is because they are trying to mitigate excessive army attrition. In this case, expected losses for both the defender and the attacker went down, as did the probability of success. The player would have to decide whether or not the slightly lowered expected losses outweigh the lowered probability of success. These results are listed in Table 11. Additionally, the strategy was graphed and compared to the optimal state-based strategies (see Figure 23).

	Markov Chain	Monte Carlo
P[Attacker Wins]:	0.8719	0.8263 +/- 0.0045
E[Attacker End State]:	6.12	6.2881 +/- 0.0391
E[Attacker Losses]:	3.88	3.7119 +/- 0.0391
E[Defender End State]:	0.32	0.4960 +/- 0.0183
E[Defender Losses]:	4.68	4.5040 +/- 0.0183
E[D Losses]:E[A Losses]:	1.2056	1.2134
E[Number of Battles]:	4.75	4.5404 +/- 0.0188

Table 11 - Case 2 Results

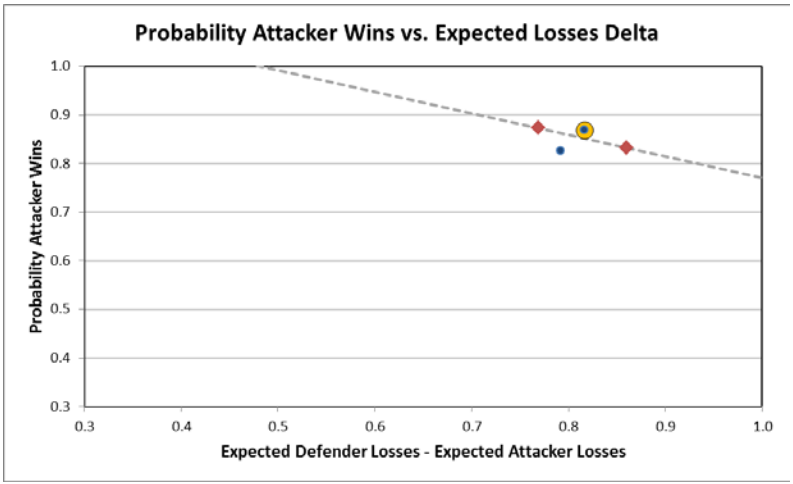

Figure 23 - Case 2 Strategy Graph

3. Case 3

		Attacker	Defender
Case 3	Start with:	10	5
	Fight with:	Maximum available (will not allow 1 vs 2)	Maximum Available
	Fight until:	2 consecutive losses	
	Win criteria:	Defender annihilation	Attacker breaks off

In this case, the expected losses were once again decreased. When compared to the Markov chain analysis, the expected end state for the defender was greatly increased (see Table 12); however, the strategy graph (see Figure 24) indicates that this policy is suboptimal.

	Markov Chain	Monte Carlo
P[Attacker Wins]:	0.8719	0.7537 +/- 0.0079
E[Attacker End State]:	6.12	6.6371 +/- 0.0309
E[Attacker Losses]:	3.88	3.3629 +/- 0.0309
E[Defender End State]:	0.32	0.8730 +/- 0.0315
E[Defender Losses]:	4.68	4.1270 +/- 0.0315
E[D Losses]:E[A Losses]:	1.2056	1.2272
E[Number of Battles]:	4.75	4.1464 +/- 0.0201

Table 12 - Case 3 Results

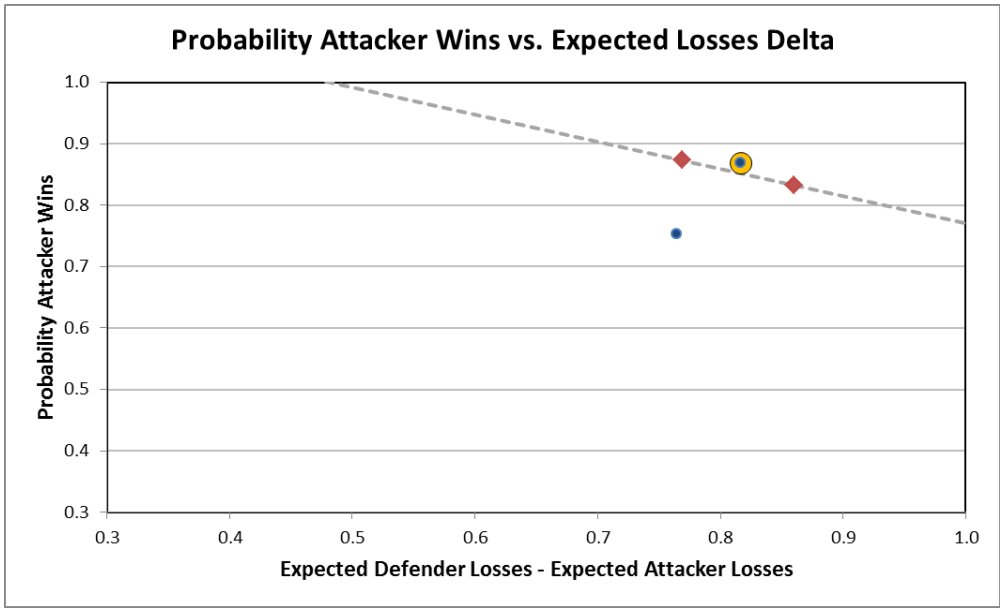

Figure 24 – Case 3 Strategy Graph

4. Case 4

		Attacker	Defender
Case 4	Start with:	10	5
	Fight with:	Maximum available (will not allow 1 vs 2)	Maximum Available
	Fight until:	3 total armies lost	
	Win criteria:	Defender annihilation	Attacker breaks off

If the attacker chooses to apply this strategy, there will be little chance for success. The probability of winning drops from 87.2% to 36.0%. Although the attacker losses have been minimized, as compared to the other three cases, the expected end state

of the defender is almost two full armies, which means this is not a good strategy if the attacker wishes to annihilate his opponent. These results are below:

	Markov Chain	Monte Carlo
P[Attacker Wins]:	0.8719	0.3600 +/- 0.0116
E[Attacker End State]:	6.12	7.4804 +/- 0.0325
E[Attacker Losses]:	3.88	2.5196 +/- 0.0325
E[Defender End State]:	0.32	1.9340 +/- 0.0274
E[Defender Losses]:	4.68	3.0660 +/- 0.0274
E[D Losses]:E[A Losses]:	1.2056	1.2169
E[Number of Battles]:	4.75	2.9946 +/- 0.0132

Table 13 - Case 4 Results

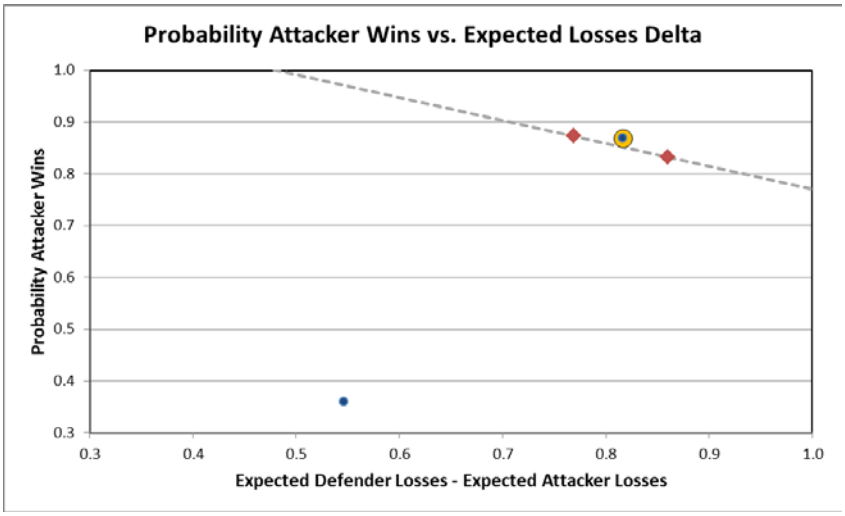

Figure 25 - Case 4 Strategy Graph

Article Comparison

The final strategy comparison was not one of the originally proposed cases. This comparison looks at the strategy policy determined to be optimal by Maliphant and Smith (1990). They determined that a policy based on an attacker ceasing engagements if his or her force strength dropped below the strength of the defender was optimized when the attacker chose to attack with one army even if when the defender used two armies.

However, by looking at this strategy graphically, it is apparent that this is a suboptimal policy (see Figure 26).

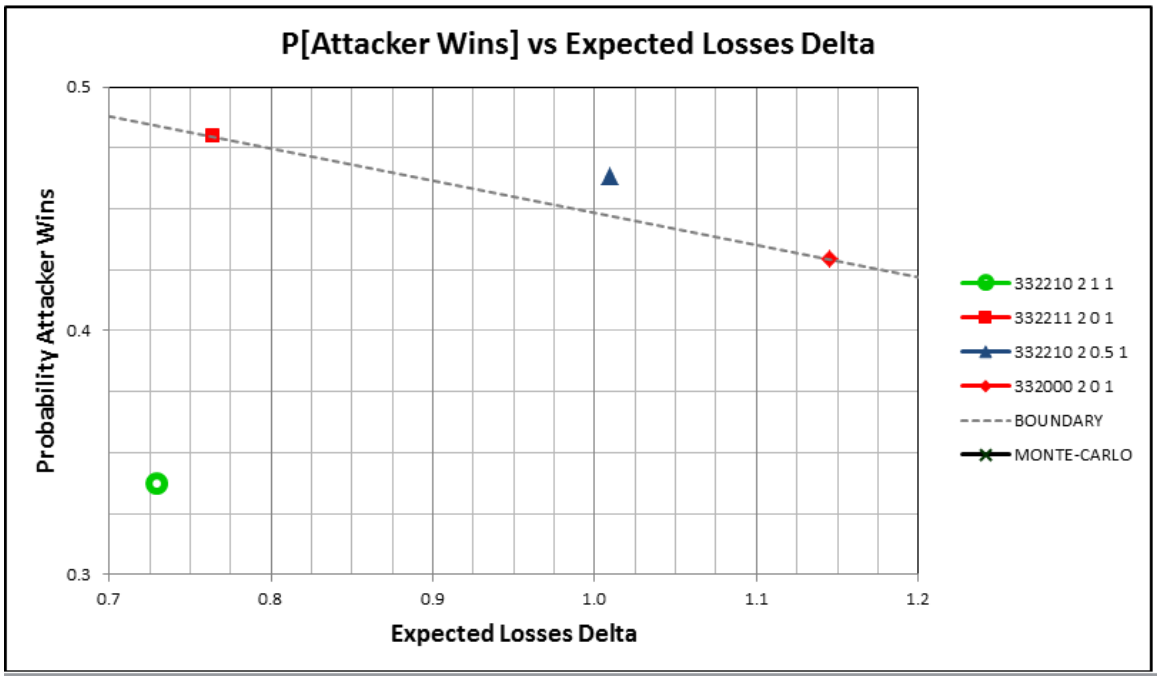

Figure 26 - Article Strategy Comparison

Determining Number of Armies

Once a strategy has been determined optimal by the player, it would also be useful to determine the number of required armies to meet an acceptable probability of winning. Using the Markovian model, the minimum number of armies can easily be calculated one of two ways.

The first way is for the user to select the appropriate strategy and initial number of defenders. The program will then provide a graphical solution to the minimum number of armies required to meet the specified acceptable probability of winning. An example of this calculation, based on a 332210 strategy and N_D = 10, is displayed in Figure 27.

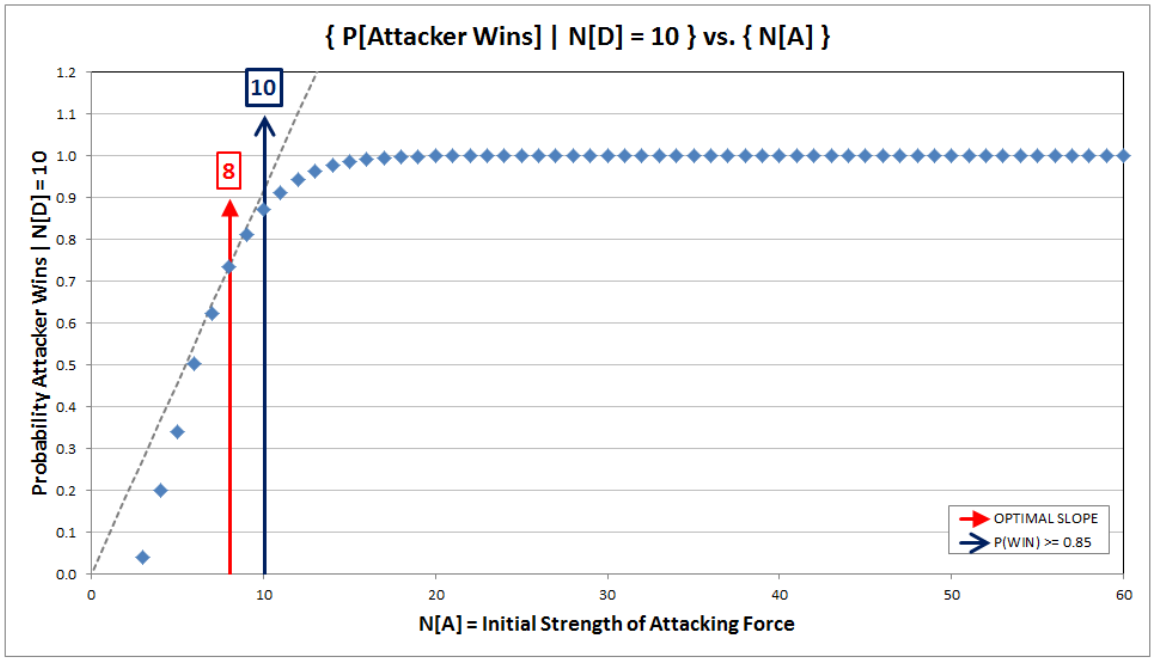

Figure 27 - Initial Force Strength Graph

The second method involves generating a table of all possible initial battle conditions, although the model is currently limited to a maximum of N_A = 60 to N_D = 30. The program then pulls the minimum number of armies required to meet or exceed the specified probability of winning.

These values are then entered into a matrix and a regression is run utilizing the least squares method, where the estimated coefficients are found by:

$$\hat{x} = (A^T A)^{-1} A^T b$$

where $\hat{x}$ is the estimated coefficients,

A is a matrix predictor variables,

b is a matrix of response variables (Strang, 2006).

A player could use the resultant coefficients as a rule of thumb to calculate the minimum number of armies required. For instance, using a 332210 strategy and a minimum probability of winning of .85, a player would need approximately five more armies than his or opponent plus and additional army for every ten defender armies. This calculation is much less than the two to three times as many attackers to defender ratio that is commonly used (see Figure 28).

N[D]	N[A]	Approximation
1	4	
2	6	
3	8	8.20
4	9	9.30
5	10	10.40
6	11	11.50
7	12	12.60
8	14	13.70
9	15	14.80
10	16	15.90
11	17	17.00
12	18	18.10
13	19	19.20
14	20	20.30
15	21	21.40
16	22	22.50
17	23	23.60
18	24	24.70
19	25	25.80
20	27	26.90
21	28	28.00
22	29	29.10
23	30	30.20
24	31	31.30
25	32	32.40
26	33	33.50
27	34	34.60
28	35	35.70
29	36	36.80
30	37	37.90

P(Attacker Wins) ≥ 0.85
Selected Strategy: 332210 2 0 1
Least-Squares Fit (N[D] ≥ 3): N[A] = 4.8506 + 1.0826 * N[D]
Approximate Fit (N[D] ≥ 3): N[A] = 4.9 + 1.10 * N[D]

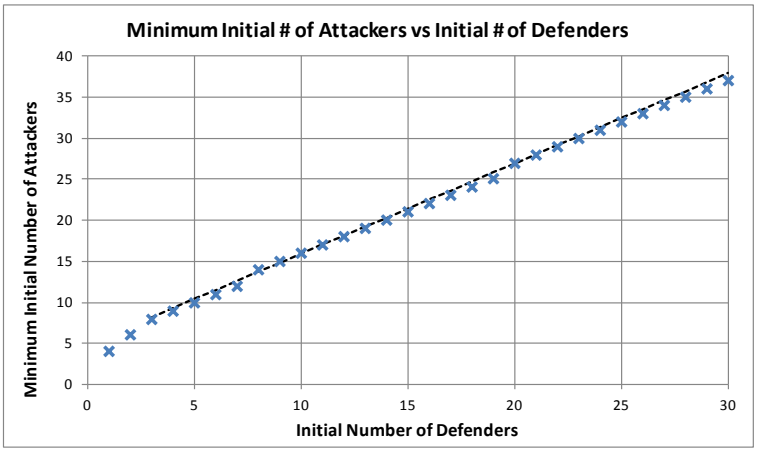

Figure 28 - Least Squares Solution

V. Conclusion

This research project looked at a broad range of methods to optimize tactical and operational strategies. By using the Markovian model, the quality of the decision is independent from the quality of the outcome, thus the policy maker is able to make confident decision even if undesirable outcomes occur. Hopefully, this prevents the decision maker from second guessing his or her decisions.

Additionally, maximizing the expected losses delta may be a superior strategy over simply looking at the probability of winning. In any case, players should consider how attrition rates are affected by the chosen strategies.

Finally, analyzing strategies through mathematical modeling can help teach the principles of strategic thinking and decision analysis. A study like this one would be easily accommodated within a class room setting. For the students with strong mathematical backgrounds, diagramming the Markov chain would be an effective teaching tool. However, regardless of the mathematically literacy of the students, a model such as the one created during this project would easily aid in classroom discussion.

References

Banks, J., Carson, J., Nelson, B., and Nicol, D. (2010). *Discrete Event System Simulation*. Prentice Hall.

Boaz, M. (2006). *Mathematical Methods in the Physical Sciences.* DePaul University.

Blatt, S. (2002). RISKy business: An in-depth look at the gameRISK. *Undergraduate Math Journal, 3.*

Carlin, B. P., & Chib, S. (1995). Bayesian model choice via markov chain monte carlo methods. *Journal of the Royal Statistical Society.Series B (Methodological),* , 473-484.

David K., S. (2007). Dynamic programming and board games: A survey. *European Journal of Operational Research, 176*(3), 1299-1318.

Georgiou, H. (2004). RISK board game-battle outcome analysis.

Griffith, S. B., & Tzu, S. (1994). The art of war. *Trans (London, Oxford University Press, 1963).*

Harju, M. (2012). On probabilities of risk type board game combats.

Herman, M., Frost, M., & Kurz, R. (2009). *Wargaming for leaders.*

Honary, E. (2007). *Total diplomacy: The art of winning risk.*

Koole, G. (1994). An optimal dice rolling policy for risk. *Nieuw Archief Voor Wiskunde, 12*(1-2), 49–52.

L'ecuyer, P. (1999). Good parameters and implementations for combined multiple recursive random number generators. *Operations Research, ,* 159-164.

Maliphant, S. A., & Smith, D. K. (1990). Mini-risk: Strategies for a simplified board game. *Journal of the Operational Research Society, Vol 4* , 9-16.

Osborne, J. A. (2003). Markov chains for the risk board game revisited. *Mathematics Magazine, 76*(2), 129.

Parker Brothers (1959). *The Rules of RISK,* Salem MA.

St-Pierre, D. D. L. (2010). *2010 IEEE symposium on computational intelligence and games.*

Strang, G. (2006). Linear Algebra and Its Applications. Brookes/Cole, Belmont CA.

Tan, B. (1997). Markov chains and the RISK board game. *Mathematics Magazine, 70*(5), 349. 0

Von Neumann, J., Morgenstern, O., Rubinstein, A., & Kuhn, H. W. (2007). *Theory of games and economic behavior.* Princeton Univ Pr.

Winston, W. L., & Goldberg, J. B. (1987). *Operations research: Applications and algorithms.* Duxbury press.

Appendix I – Furthest Optimal Strategy from LaGrange Boundary

A \ D	30	29	28	27	26	25	24	23	22	21
60	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1
59	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
58	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1
57	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
56	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
55	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
54	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
53	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
52	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
51	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
50	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
49	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
48	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
47	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
46	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
45	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
44	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
43	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
42	332210 2 0.25 1	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
41	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
40	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
39	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
38	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
37	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
36	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
35	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1
34	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
33	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1
32	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1
31	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1
30	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1
29	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
28	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1
27	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
26	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
25	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
24	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
23	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
22	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
21	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
20	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
19	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
18	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
17	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
16	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
15	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
14	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
13	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
12	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
11	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
10	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
9	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1
8	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1
7	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1
6	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1
5	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1
4	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1
3	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2
2	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1

A \ D	20	19	18	17	16	15	14	13	12	11
60	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
59	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
58	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
57	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
56	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
55	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
54	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
53	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
52	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
51	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
50	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
49	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
48	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
47	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
46	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
45	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
44	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
43	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
42	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
41	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
40	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
39	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
38	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
37	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
36	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
35	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
34	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
33	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
32	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
31	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
30	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1
29	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
28	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1
27	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1
26	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.75 1
25	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332010 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1
24	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1
23	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332010 2 0 1
22	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
21	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
20	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
19	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
18	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
17	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
16	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
15	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1
14	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1
13	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0.5 1
12	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1	332210 2 0.5 1
11	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.5 1	332210 2 0 1
10	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1	332210 2 0.5 1
9	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1
8	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1	332210 2 0 1
7	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1
6	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1	332210 2 0 1
5	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1
4	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0.25 1	332210 2 0 1	332210 2 0 1
3	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2
2	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1

A \ D	10	9	8	7	6	5	4	3	2	1
60	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 1 1	332210 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1
59	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332210 2 0 1	332211 2 0 1	332211 2 0 1
58	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 1 1	332210 2 0 1	332210 2 1.5 1	332211 2 0 1	332211 2 0 1
57	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332210 2 0 1	332211 2 0 1	332211 2 0 1
56	332010 2 0 1	332210 2 0.75 1	332010 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 0 1	332211 2 0 1
55	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332210 2 0 1	332211 2 1.5 1	332211 2 0 1
54	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 0 2	332211 2 0 1
53	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
52	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
51	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
50	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
49	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
48	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
47	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
46	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
45	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
44	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
43	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
42	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
41	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
40	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
39	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
38	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
37	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
36	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
35	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
34	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
33	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 2
32	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
31	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
30	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
29	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
28	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
27	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
26	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
25	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
24	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
23	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
22	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
21	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
20	332210 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
19	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
18	332210 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
17	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
16	332210 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
15	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332010 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
14	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332210 2 0.75 1	332010 2 0 1	332010 2 0 1	332211 2 1.5 1	332211 2 0 1
13	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
12	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332211 2 0 1
11	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
10	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0 1	332210 2 0.75 1	332010 2 0 1	332211 2 1 1	332211 2 1.5 1	332211 2 0 1
9	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
8	332210 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0 1	332210 2 0.75 1	332210 2 0 1	332211 2 1 1	332211 2 1.5 1	332211 2 0 1
7	332210 2 0 1	332210 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332210 2 0.75 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
6	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.5 1	332210 2 0 1	332210 2 0.75 1	332210 2 0 1	332211 2 1 1	332211 2 1.5 1	332211 2 0 1
5	332210 2 0 1	332210 2 0 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.5 1	332210 2 0.75 1	332211 2 1 1	332211 2 1.5 1	332210 2 1.5 1	332211 2 0 1
4	332210 2 0 1	332210 2 0 1	332210 2 0 1	332210 2 0.5 1	332210 2 0 1	332210 2 0.75 1	332210 2 0 1	332211 2 1 1	332211 2 1.5 1	332211 2 0 1
3	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 2	332211 2 0 1
2	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1	332211 2 0 1

Appendix II – Markov Chain VBA Code

```
'This subroutine builds a Markov Chain (P-, N-, and A-Matrices):
Sub Build_Markov_Chain()

    'Declaration of local variables
    Dim i As Integer                    'counter variable
    Dim j As Integer                    'counter variable
    Dim k As Integer                    'counter variable
    Dim iA As Integer                   'current number of attackers
    Dim iD As Integer                   'current number of defenders

    Dim iForce_EndState As Integer      'the current attacker's end state based on a ratio
                                        '...between attacker and defender (minus a const)

    Dim iEndState As Integer            'the highest attacker end state based on the
                                        '...current conditions; maximum of force ratio and
                                        '...minimum number of required remaining armies

    Dim iRow As Integer                 'the row of the associated absorbing state
    Dim iColumn As Integer              'the column of all losing absorbing states
    Dim Header_Matrix() As Integer      'binary array indicating win or loss
    Dim Column_Matrix() As Integer      'array of attacker and defender end states

    'Initialization of local variables
    k = 0
    iA = iAttacker_Initial
    iD = iDefender_Initial
    iAbsorb_State_Win = 0

    'Resize the Matrix_Build array to [A x D] dimensions
    ReDim Matrix_Build(iDefender_Initial + 1, iAttacker_Initial)

    'First entry into the Matrix_Build array is initial attacker vs initial defender
    Matrix_Build(1, 1) = Format_State(iA, iD)

    'Calculates all possible states based on the associated decrements
    For i = 1 To iDefender_Initial + 1
        For j = 1 To iAttacker_Initial
            If Matrix_Build(i, j) <> "" Then
                k = k + 1 'counter of total states
                If i = iDefender_Initial + 1 Then iAbsorb_State_Win = iAbsorb_State_Win + 1
                iA = Left(Matrix_Build(i, j), 2)
                iD = Right(Matrix_Build(i, j), 2)

                'Determines current end state
```

```
            iForce_EndState = Int(iD * dForce_Ratio) - iForce_Ratio_Decrement
            iEndState = Ap.Max(iAttacker_EndState, iForce_EndState)

            'Determines current engagement condition and builds possible states
            If iA - iEndState >= 2 Then
               If iA >= 4 And iD >= 2 Then
                  Call Build_States(iA, iD, 3, 2)
               ElseIf iA >= 4 And iD = 1 Then
                  Call Build_States(iA, iD, 3, 1)
               ElseIf iA = 3 And iD >= 2 Then
                  Call Build_States(iA, iD, 2, 2)
               ElseIf iA = 3 And iD = 1 Then
                  Call Build_States(iA, iD, 2, 1)
               ElseIf iD = 0 Then
               Else: MsgBox "ERROR: ln 59"
               End If
            ElseIf iA - iEndState = 1 Then
               If iA >= 2 And iD >= 2 Then
                  Call Build_States(iA, iD, 1, 2)
               ElseIf iA >= 2 And iD = 1 Then
                  Call Build_States(iA, iD, 1, 1)
               ElseIf iA = 1 Or iD = 0 Then
               Else: MsgBox "ERROR: line 67"
               End If
            End If

         End If

      Next

   Next

   'Adjusts for initial losses (if specified)
   For j = 1 To iAttacker_Initial
      If Matrix_Build(1, j) <> "" Then
         If Int(Left(Matrix_Build(1, j), 2)) < iInitial_Losses Then
            Matrix_Build(1, j) = ""
            k = k - 1
         End If
      End If
   Next

   'Resizes the P-Matrix headers based on total number of states
   ReDim P_Matrix_Headers(k, 1)

   'Transfer the winning absorption states from the Matrix_Build array to the
```

```
'...P_Matrix_Headers array
j = 0
For i = k - (iAbsorb_State_Win - 1) To k
   j = j + 1
   P_Matrix_Headers(i, 1) = Matrix_Build(iDefender_Initial + 1, j)
Next

'Finds the losing absorbing state column in the Matrix_Build array
iColumn = iAttacker_Initial
iAbsorb_State_Lose = 0
Do While iAbsorb_State_Lose = 0
   For i = 1 To iDefender_Initial
      If Matrix_Build(i, iColumn) <> "" Then iAbsorb_State_Lose = 1
   Next
   If iAbsorb_State_Lose = 0 Then iColumn = iColumn - 1
Loop

'If there is no absorbing state in the absorbing state column for a particular numnber
'...defenders, then pull one from an adjacent column and transfer it to the
'...P_Matrix_Headers array
For iRow = 1 To iDefender_Initial
   If Matrix_Build(iRow, iColumn) = "" Then
      iAbsorb_State_Lose = 0
      j = iColumn
      Do While iAbsorb_State_Lose = 0
         j = j - 1
         If Matrix_Build(iRow, j) <> "" Then
            P_Matrix_Headers(k - (iAbsorb_State_Win + iDefender_Initial) + _
               iRow, 1) = Matrix_Build(iRow, j)
            Matrix_Build(iRow, j) = ""
            iAbsorb_State_Lose = 1
         End If
      Loop
   Else
      P_Matrix_Headers(k - (iAbsorb_State_Win + iDefender_Initial) + iRow, 1) _
         = Matrix_Build(iRow, iColumn)
      Matrix_Build(iRow, iColumn) = ""
   End If
Next

'Transfer all transient states to the P_Matrix_Headers array
iRow = 0
For j = 1 To iColumn - 1
   For i = 1 To iDefender_Initial
      If Matrix_Build(i, j) <> "" Then
         iRow = iRow + 1
```

```
            P_Matrix_Headers(iRow, 1) = Matrix_Build(i, j)
        End If
    Next
Next

'Calculate number of states (absorption, transient, and total)
iAbsorb_State_Lose = iDefender_Initial
iAbsorb_State_Total = iAbsorb_State_Win + iAbsorb_State_Lose
iTransient_State = k - iAbsorb_State_Total
iTotal_State = k

'Resizes Matrix_Build array to save computer memory
ReDim Matrix_Build(1, 1)

'Resizes P-, Q- , R- and N- matrices based on the number of states
ReDim P_Matrix(iTotal_State, iTotal_State)
ReDim Q_Matrix(iTransient_State, iTransient_State)
ReDim R_Matrix(iTransient_State, iAbsorb_State_Total)
ReDim N_Matrix(iTransient_State, iTransient_State)

'Determines current engagement condition and calculates transition probabilities;
'...builds P-, Q-, and R- matrices
For i = 1 To iTransient_State
    iA = Left(P_Matrix_Headers(i, 1), 2)
    iD = Right(P_Matrix_Headers(i, 1), 2)
    iForce_EndState = Int(iD * dForce_Ratio) - iForce_Ratio_Decrement
    iEndState = Ap.Max(iAttacker_EndState, iForce_EndState)
    If iA - iEndState >= 2 Then     'Attacker can use more than one army
        If iA >= 4 And iD >= 2 Then     '3 vs 2 scenario
            Call Markov_Prob(iA, iD, i, 3, 2)
        ElseIf iA >= 4 And iD = 1 Then  '3 vs 1 scenario
            Call Markov_Prob(iA, iD, i, 3, 1)
        ElseIf iA = 3 And iD >= 2 Then  '2 vs 2 scenario
            Call Markov_Prob(iA, iD, i, 2, 2)
        ElseIf iA = 3 And iD = 1 Then   '2 vs 1 scenario
            Call Markov_Prob(iA, iD, i, 2, 1)
        End If
    ElseIf iA - iEndState = 1 Then  'Attacker can only lose one army

        'If defender chooses to defend with only army (even though more may be
        '...available and the attacker has chosen a dependent strategy policy,
        '...then the attacker can attack with three armies and still only
        '...risk losing one army
        If iA >= 3 And iD >= 2 Then
            If iDefender_Strategy(2) = 1 And blnIndependent_Strategy = False Then
                Call Markov_Prob(iA, iD, i, 3, 1)
```

```
            Else: Call Markov_Prob(iA, iD, i, 1, 2) 'otherwise, 1 vs 2 scenario
            End If

        ElseIf iA >= 4 And iD = 1 Then  '3 vs 1 scenario
            Call Markov_Prob(iA, iD, i, 3, 1)

        'If defender chooses to defend with only army (even though more may be
        '...available and the attacker has chosen a dependent strategy policy,
        '...then the attacker can attack with two armies and still only
        '...risk losing one army
        ElseIf iA = 3 And iD >= 2 Then
            If iDefender_Strategy(2) = 1 And blnIndependent_Strategy = False Then
                Call Markov_Prob(iA, iD, i, 2, 1)
            Else: Call Markov_Prob(iA, iD, i, 1, 2) 'otherwise, 1 vs 2 scenario
            End If

        ElseIf iA = 3 And iD = 1 Then   '2 vs 1 scenario
            Call Markov_Prob(iA, iD, i, 2, 1)
        ElseIf iA = 2 And iD >= 2 Then  '1 vs 2 scenario
            Call Markov_Prob(iA, iD, i, 1, 2)
        ElseIf iA = 2 And iD = 1 Then   '1 vs 1 scenario
            Call Markov_Prob(iA, iD, i, 1, 1)
        End If
    End If
Next

'Fills in the 0- and I- submatrix portions of the P-Matrix
For i = iTransient_State + 1 To iTotal_State
    For j = 1 To iTotal_State
        If i = j Then
            P_Matrix(i, j) = 1
        Else: P_Matrix(i, j) = 0
        End If
    Next
Next

'N-Matrix at this point = I-Q; to complete the calculation N is inverted
N_Matrix = Ap.MInverse(N_Matrix)

'Calculates absorption matrix
A_Matrix = Ap.MMult(N_Matrix, R_Matrix)

'Creates a binary win-loss array
ReDim Header_Matrix(iAbsorb_State_Total, 2)
For i = 1 To iAbsorb_State_Total
    If Right(P_Matrix_Headers(iTransient_State + i, 1), 2) = 0 Then
```

```
            Header_Matrix(i, 1) = 1
            Header_Matrix(i, 2) = 0
        Else
            Header_Matrix(i, 1) = 0
            Header_Matrix(i, 2) = 1
        End If
    Next

    'Calculates the probability of winning/losing for a given initial condition
    Win_Lose_Matrix = Ap.MMult(A_Matrix, Header_Matrix)

    'Creates an array of the attacker and defender end states
    ReDim Column_Matrix(iAbsorb_State_Total, 2)
    For i = 1 To iAbsorb_State_Total
        Column_Matrix(i, 1) = Left(P_Matrix_Headers(iTransient_State + i, 1), 2)
        Column_Matrix(i, 2) = Right(P_Matrix_Headers(iTransient_State + i, 1), 2)
    Next

    'Calculates expected end states
    Exp_End_State_Matrix = Ap.MMult(A_Matrix, Column_Matrix)

    'Creates a matrix with expected losses values are squared (will be used to find
variance)
    ReDim Preserve Column_Matrix(iAbsorb_State_Total, 3)
    For i = 1 To iAbsorb_State_Total
        Column_Matrix(i, 3) = ((iDefender_Initial - Column_Matrix(i, 2)) - _
            (iAttacker_Initial - Column_Matrix(i, 1))) ^ 2
        Column_Matrix(i, 1) = (iAttacker_Initial - Column_Matrix(i, 1)) ^ 2
        Column_Matrix(i, 2) = (iDefender_Initial - Column_Matrix(i, 2)) ^ 2
    Next

    'Caluclates the 2nd moment of the expected losses (only for the initial set of
conditions)
    Exp_Losses_2ndMoment = Ap.MMult(Ap.Index(A_Matrix, 1, 0), Column_Matrix)

    'Resize expected losses matrix
    ReDim Exp_Losses_Matrix(iTransient_State, 3)

    'Calculates expected losses matrix
    If iTransient_State = 0 Then
        MsgBox "There are no transient states.  Please check initial conditions"
    ElseIf iTransient_State = 1 Then
        Exp_Losses_Matrix(1, 1) = Left(P_Matrix_Headers(1, 1), 2) -
Exp_End_State_Matrix(1)
        Exp_Losses_Matrix(1, 2) = Right(P_Matrix_Headers(1, 1), 2) -
Exp_End_State_Matrix(2)
```

```
      Exp_Losses_Matrix(1, 3) = Exp_Losses_Matrix(1, 2) - Exp_Losses_Matrix(1, 1)
   Else
      For i = 1 To iTransient_State
         Exp_Losses_Matrix(i, 1) = Left(P_Matrix_Headers(i, 1), 2) - _
            Exp_End_State_Matrix(i, 1)
         Exp_Losses_Matrix(i, 2) = Right(P_Matrix_Headers(i, 1), 2) - _
            Exp_End_State_Matrix(i, 2)
         Exp_Losses_Matrix(i, 3) = Exp_Losses_Matrix(i, 2) - Exp_Losses_Matrix(i, 1)
      Next
   End If

   'Calculates conditional expected losses (E[Losses] | A Wins or D Wins)
   ReDim Column_Matrix(iAbsorb_State_Total, 2)
   For i = 1 To iAbsorb_State_Total
      Column_Matrix(i, 1) = ((iDefender_Initial - _
         Right(P_Matrix_Headers(iTransient_State + i, 1), 2)) - _
         (iAttacker_Initial - Left(P_Matrix_Headers(iTransient_State + i, 1), 2))) * _
         Header_Matrix(i, 1)
      Column_Matrix(i, 2) = ((iDefender_Initial - _
         Right(P_Matrix_Headers(iTransient_State + i, 1), 2)) - _
         (iAttacker_Initial - Left(P_Matrix_Headers(iTransient_State + i, 1), 2))) * _
         Header_Matrix(i, 2)
   Next
   Exp_Conditional_Delta = Ap.MMult(Ap.Index(A_Matrix, 1, 0), Column_Matrix)

End Sub
'*************************************************************************
************************
'This subroutine calculates the subsequent possible states based on a current state:
Sub Build_States(iA As Integer, iD As Integer, i As Integer, j As Integer)

   'Declaration of local variables
   Dim k As Integer                 'counter variable
   Dim iA_Next As Integer           'next attacker state
   Dim iD_Next As Integer           'next defender state

   'Builds the Matrix_Build array
   For k = 0 To iDecrement(i, j)
      iA_Next = iA - k
      iD_Next = iD + k - iDecrement(i, j)
      Matrix_Build(iDefender_Initial - iD_Next + 1, iAttacker_Initial - iA_Next + 1) = _
         Format_State(iA_Next, iD_Next)
   Next

End Sub
'*************************************************************************
```

```
'This subroutine enters the transtition probabilities into the P-, Q-, R-, and (I-Q) matrices:
Sub Markov_Prob(iA As Integer, iD As Integer, i As Integer, X As Integer, Y As Integer)

   'Declaration of local counter variable
   Dim j As Integer

   If iDecrement(X, Y) = 2 Then    'total attrition = 2 armies (i.e. ties are possible)
      For j = 1 To iTotal_State

         'Attacker wins: defender loses two armies
         If iA = Left(P_Matrix_Headers(j, 1), 2) _
           And iD = Right(P_Matrix_Headers(j, 1), 2) + 2 Then
            P_Matrix(i, j) = Prob_Win(X, Y)
            If j <= iTransient_State Then
               Q_Matrix(i, j) = Prob_Win(X, Y)
               N_Matrix(i, j) = 0 - Prob_Win(X, Y)
            Else: R_Matrix(i, j - iTransient_State) = Prob_Win(X, Y)
            End If

         'Tie: both sides lose one army
         ElseIf iA = Left(P_Matrix_Headers(j, 1), 2) + 1 _
           And iD = Right(P_Matrix_Headers(j, 1), 2) + 1 Then
            P_Matrix(i, j) = Prob_Tie(X, Y)
            If j <= iTransient_State Then
               Q_Matrix(i, j) = Prob_Tie(X, Y)
               N_Matrix(i, j) = 0 - Prob_Tie(X, Y)
            Else: R_Matrix(i, j - iTransient_State) = Prob_Tie(X, Y)
            End If

         'Defender wins: attacker loses two armies
         ElseIf iA = Left(P_Matrix_Headers(j, 1), 2) + 2 _
           And iD = Right(P_Matrix_Headers(j, 1), 2) Then
            P_Matrix(i, j) = Prob_Lose(X, Y)
            If j <= iTransient_State Then
               Q_Matrix(i, j) = Prob_Lose(X, Y)
               N_Matrix(i, j) = 0 - Prob_Lose(X, Y)
            Else: R_Matrix(i, j - iTransient_State) = Prob_Lose(X, Y)
            End If

         'Otherwise, the state is not reachable from the current state;
         Else
            P_Matrix(i, j) = 0
            If j <= iTransient_State Then
               Q_Matrix(i, j) = 0
               If i = j Then
                  N_Matrix(i, j) = 1
```

```
                Else: N_Matrix(i, j) = 0
                End If
            Else: R_Matrix(i, j - iTransient_State) = 0
            End If
        End If
    Next

ElseIf iDecrement(X, Y) = 1 Then    'total attrition = 1 army (i.e. ties are not possible)
    For j = 1 To iTotal_State

        'Attacker wins: defender loses one army
        If iA = Left(P_Matrix_Headers(j, 1), 2) _
          And iD = Right(P_Matrix_Headers(j, 1), 2) + 1 Then
            P_Matrix(i, j) = Prob_Win(X, Y)
            If j <= iTransient_State Then
                Q_Matrix(i, j) = Prob_Win(X, Y)
                N_Matrix(i, j) = 0 - Prob_Win(X, Y)
            Else: R_Matrix(i, j - iTransient_State) = Prob_Win(X, Y)
            End If

        'Defender wins: attacker loses one army
        ElseIf iA = Left(P_Matrix_Headers(j, 1), 2) + 1 _
          And iD = Right(P_Matrix_Headers(j, 1), 2) Then
            P_Matrix(i, j) = Prob_Lose(X, Y)
            If j <= iTransient_State Then
                Q_Matrix(i, j) = Prob_Lose(X, Y)
                N_Matrix(i, j) = 0 - Prob_Lose(X, Y)
            Else: R_Matrix(i, j - iTransient_State) = Prob_Lose(X, Y)
            End If

        'Otherwise, the state is not reachable from the current state;
        '...fill in the element with zero
        Else
            P_Matrix(i, j) = 0
            If j <= iTransient_State Then
                Q_Matrix(i, j) = 0
                If i = j Then
                    N_Matrix(i, j) = 1
                Else: N_Matrix(i, j) = 0
                End If
            Else: R_Matrix(i, j - iTransient_State) = 0
            End If
        End IF
    Next
End IF
End Sub
```

Appendix III – Monte Carlo Simulation VBA Code

```
'This subroutine runs a discrete event simulation using Monte-Carlo methods for
generating the state-change events (i.e. rolling the dice):
Sub Run_Monte_Carlo()
    'Seeds the MRG32k5a random number generator
    Call RNG_Seed
    'Declaration of local variables
    Dim i As Integer                    'counter variable
    Dim j As Integer                    'counter variable
    Dim k As Integer                    'counter variable
    Dim N As Integer                    'counter variable (number of engagements)
    Dim iA As Integer                   'current number of attackers
    Dim iD As Integer                   'current number of defenders

    Dim iForce_EndState As Integer      'the current attacker's end state based on a ratio
                                        '...between attacker and defender (minus a const)

    Dim iEndState As Integer            'the highest attacker end state based on the
                                        '...current conditions; maximum of force ratio and
                                        '...minimum number of required remaining armies

    Dim datTime As Date                 'timer variable; used to control animation speed

    Dim blnContinue As Boolean          'logic variable; used to determine if absorption
                                        '...state is reached

    Dim iState() As Variant             'tracks current state of the battle

    Dim Rep_Stats() As Variant          'records statistics for current batch
    Dim Batch_Stats() As Variant        'records summary statistics for all batches
    Dim Absorb_States() As Double       'tallies absorbing tracks
    Dim strA As String                  'outputs the number of attacker armies (animation)
    Dim strD As String                  'outputs the number of defender armies (animation)
    Dim response As String              'determines if simulation should continue running

    'Ensures "Sim Animation" worksheet is selected and scaled
    Application.ScreenUpdating = False
    Call View_Sheet("Sim Animation", True)
    ActiveWindow.WindowState = xlMaximized
    Range("A1:I10").Select
    ActiveWindow.Zoom = True

    'Clears contents and selected formatting
    Call Animate_Clear

    'Ensures "Data Sheet" is accessible for updating graph data (animation)
```

```
Sheets("Data Sheet").Visible = True

'Updates screen with empty battle field
Application.ScreenUpdating = True
Application.ScreenUpdating = False

'System delay
datTime = Timer + 1
Do
   DoEvents
Loop Until Timer >= datTime

'Initializes the force strength for both sides (animation)
'...Note: each character "x" represents one army
For i = 1 To iAttacker_Initial
   strA = strA & "x"
Next
For i = 1 To iDefender_Initial
   strD = strD & "x"
Next

'Resizes statistic arrays for the selected number of replications and batches
ReDim Batch_Stats(iBatch, 8)
ReDim Rep_Stats(iReplications, 8)

'Resizes the absorption state array for the initial conditions
ReDim Absorb_States(iAttacker_Initial, iDefender_Initial + 1)

'-------- BATCH --------
For j = 1 To iBatch

   '-------- REPLICATION --------
   For i = 1 To iReplications

   'Initializes the force strength for both sides
   iA = iAttacker_Initial
   iD = iDefender_Initial

   'The battle will continue as long as blnContinue = True
   blnContinue = True

   'Initializes additional cutoff criteria counters
   iCum_Loss(1) = 0
   iCon_Loss(1) = 0
   iCum_Armies(1) = 0
   iCon_Armies(1) = 0
```

```
        'Initializes the number of engagements counter
        N = 0

        'Update replication/batch counter
        Range("B2") = "Run: " & ((j - 1) * iReplications) + i & "/" & _
           (iReplications * iBatch)
        Range("B2").Select
        Application.ScreenUpdating = True
        Application.ScreenUpdating = False

        'Initializes battle animation if animation is desired
        If Range("rngAnimate") = True Then
           blnAnimate = True
           Call Clear_Shapes("Sim Animation")
           Range("B4") = strA
           Range("H4") = strD
           Range("E2") = iAttacker_Initial & " Attackers vs " & _
              iDefender_Initial & " Defenders"
           Range("E2").Select
           Application.ScreenUpdating = True
           Application.ScreenUpdating = False
           datTime = Timer + 1 + (Speed / 10)
           Do
              DoEvents
           Loop Until Timer >= datTime
          Range("E2").ClearContents

        'Clears screen if animation is not desired
        Else
           blnAnimate = False
           If Range("B4") <> "" Then Call Animate_Clear
        End If

        'Continue battle until cutoff criteria is achieved
        Do While blnContinue = True

           'If user terminates Sim_Run_Menu, then checks to see if user wants to stop the
sim
           If blnRunSim = False Then
              response = MsgBox("Do you want to stop the simulation?", vbYesNo,
"Confirm")
              If response = vbYes Then
                 Call Animate_Clear
                 Exit Sub
              Else
```

```
      blnRunSim = True
      Sim_Run_Menu.Show
   End If
End If


If blnAnimate = True Then
   Call Clear_Shapes("Sim Animation")
   Range("B4") = Left(strA, iA)
   Range("H4") = Left(strA, iD)
End If

iForce_EndState = Int(iD * dForce_Ratio) - iForce_Ratio_Decrement
iEndState = Ap.Max(iAttacker_EndState, iForce_EndState)
N = N + 1
If iA - iEndState >= 2 Then
   If iA >= 4 And iD >= 2 Then
      If iAttacker_Strategy(3, 2) = 0 Then
         blnContinue = False
      Else: iState = Dice_Roll(iA, iD, 3, 2)
      End If
   ElseIf iA >= 4 And iD = 1 Then
      If iAttacker_Strategy(3, 1) = 0 Then
         blnContinue = False
      Else: iState = Dice_Roll(iA, iD, 3, 1)
      End If
   ElseIf iA = 3 And iD >= 2 Then
      If iAttacker_Strategy(2, 2) = 0 Then
         blnContinue = False
      Else: iState = Dice_Roll(iA, iD, 2, 2)
      End If
   ElseIf iA = 3 And iD = 1 Then
      If iAttacker_Strategy(2, 1) = 0 Then
         blnContinue = False
      Else: iState = Dice_Roll(iA, iD, 2, 1)
      End If
   End If
ElseIf iA - iEndState = 1 Then
   If iA >= 2 And iD >= 2 Then
      If iAttacker_Strategy(1, 2) = 0 Then
         blnContinue = False
      Else: iState = Dice_Roll(iA, iD, 1, 2)
      End If
   ElseIf iA >= 4 And iD = 1 Then
      If iAttacker_Strategy(3, 1) = 0 Then
         blnContinue = False
```

```
            Else: iState = Dice_Roll(iA, iD, 3, 1)
            End If
          ElseIf iA = 3 And iD = 1 Then
            If iAttacker_Strategy(2, 1) = 0 Then
              blnContinue = False
            Else: iState = Dice_Roll(iA, iD, 2, 1)
            End If
          ElseIf iA = 2 And iD = 1 Then
            If iAttacker_Strategy(1, 1) = 0 Then
              blnContinue = False
            Else: iState = Dice_Roll(iA, iD, 1, 1)
            End If
          End If
        Else
          N = N - 1
          blnContinue = False
        End If


        iA = iState(1)
        iD = iState(2)
        If iState(3) = 0 Then
          iCon_Loss(1) = 0
        Else: iCon_Loss(1) = iCon_Loss(1) + iState(3)
        End If
        iCum_Loss(1) = iCum_Loss(1) + iState(3)
        If iState(4) = 0 Then
          iCon_Armies(1) = 0
        Else: iCon_Armies(1) = iCon_Armies(1) + iState(4)
        End If
        iCum_Armies(1) = iCum_Armies(1) + iState(4)

        If iA <= iEndState Or iD = 0 Then blnContinue = False
        If N <= iCum_Loss(3) And iCum_Loss(1) >= iCum_Loss(2) Then blnContinue =
False
        If N <= iCon_Loss(3) And iCon_Loss(1) >= iCon_Loss(2) Then blnContinue =
False
        If N <= iCum_Armies(3) And iCum_Armies(1) >= iCum_Armies(2) Then
blnContinue = False
        If N <= iCon_Armies(3) And iCon_Armies(1) >= iCon_Armies(2) Then
blnContinue = False

        DoEvents
      Loop

      If iD = 0 Then
```

```
            Rep_Stats(i, 1) = 1
            Rep_Stats(i, 3) = 0
            Rep_Stats(i, 7) = (iDefender_Initial - iD) - (iAttacker_Initial - iA)
            Rep_Stats(i, 8) = 0
            If blnAnimate = True Then Call Animate_Victory("ATTACKER WINS")
        Else
            Rep_Stats(i, 1) = 0
            Rep_Stats(i, 3) = 1
            Rep_Stats(i, 7) = 0
            Rep_Stats(i, 8) = (iDefender_Initial - iD) - (iAttacker_Initial - iA)
            If blnAnimate = True Then Call Animate_Victory("DEFENDER WINS")
        End If

        Rep_Stats(i, 2) = iA
    Rep_Stats(i, 4) = iD
    Rep_Stats(i, 5) = (iDefender_Initial - iD) - (iAttacker_Initial - iA)
    Rep_Stats(i, 6) = N
    Absorb_States(iA, iD + 1) = Absorb_States(iA, iD + 1) + 1

    Next

    For k = 1 To 8
    Batch_Stats(j, k) = Ap.Average(Ap.Index(Rep_Stats, 0, k))
    Next

Next

Sheets("Sim Results").Visible = True
Sheets("Sim Results").Select
ActiveSheet.Unprotect

Range("M7").Select
Range(Selection, Selection.End(xlToRight)).Select
Range(Selection, Selection.End(xlDown)).Select
Selection.ClearContents

'Range("A28:J500").Clear

For k = 1 To 6
    Cells(3, 13 + k) = Ap.Sum(Ap.Index(Batch_Stats, 0, k)) / iBatch
    Cells(4, 13 + k) = Ap.StDev_S(Ap.Index(Batch_Stats, 0, k))
    Cells(5, 13 + k) = Ap.Confidence_T(0.05, Cells(4, 13 + k), iBatch)
Next

'Fill in summary table
Range("F3") = Range("N3")
```

```
Range("F4") = Range("P3")
Range("F7") = Range("O3")
Range("F8") = iAttacker_Initial - Range("F7")
Range("F9") = Range("Q3")
Range("F10") = iDefender_Initial - Range("F9")
Range("F11") = Range("R3")
Range("F16") = Range("N3")
Range("F21") = Range("P3")
Range("G3") = Range("N5")
Range("G4") = Range("P5")
Range("G7") = Range("O5")
Range("G9") = Range("Q5")
Range("G11") = Range("R5")

ActiveSheet.Shapes.Range(Array("Oval 24")).Select
Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = _
   Format(Ap.Sum(Ap.Index(Batch_Stats, 0, 7)) / Ap.Sum(Ap.Index(Batch_Stats, 0, 1)),
"0.000")
ActiveSheet.Shapes.Range(Array("Oval 23")).Select
Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = _
   Format(Ap.Sum(Ap.Index(Batch_Stats, 0, 8)) / Ap.Sum(Ap.Index(Batch_Stats, 0, 3)),
"0.000")
ActiveSheet.Shapes.Range(Array("Oval 17")).Select
Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = _
   Format(Ap.Sum(Ap.Index(Batch_Stats, 0, 5)) / iBatch, "0.000")


For i = 1 To iBatch
   Cells(6 + i, 13) = i
Next

Range(Cells(7, 14), Cells(6 + iBatch, 19)) = Batch_Stats

Sheets("Strategy Graph").Visible = True
Sheets("Strategy Graph").Select
ActiveSheet.Unprotect

For i = 3 To 4
   Cells(i, 11) = Sheets("Sim Results").Cells(i, 6)
Next

For i = 7 To 11
   Cells(i, 11) = Sheets("Sim Results").Cells(i, 6)
Next

Sheets("A-Matrix").Visible = True
```

```
Sheets("A-Matrix").Select
ActiveSheet.Unprotect

Range("B2:B3").Select
Range(Selection, Selection.End(xlToRight)).Select
Selection.Copy

Sheets("Sim Results").Select
Range("A28").Select
Selection.PasteSpecial Paste:=xlPasteAll, Operation:=xlNone, SkipBlanks:= _
   False, Transpose:=True
Range("B28").Select
Range(Selection, Selection.End(xlDown)).NumberFormat = "0.0000"
Call Format_Borders(Range("A28"), 3)
Range("A28").Select
Range(Selection, Selection.End(xlDown)).Select
Range(Selection, Selection.End(xlToRight)).Select
Selection.Copy
Range("E28").Select
ActiveSheet.Paste
Range("I28").Select
ActiveSheet.Paste
Range("F28").Select
Range(Selection, Selection.End(xlDown)).ClearContents
Range("J28").Select
Range(Selection, Selection.End(xlDown)).ClearContents

i = 0
Do While Range("E28").Offset(i, 0) <> ""
   iA = Left(Range("E28").Offset(i, 0), 2)
   iD = Right(Range("E28").Offset(i, 0), 2)
   Range("E28").Offset(i, 1) = Absorb_States(iA, iD + 1) / (iBatch * iReplications)
   Range("E28").Offset(i, 1).NumberFormat = "0.0000"
   Absorb_States(iA, iD + 1) = 0
   i = i + 1
Loop

   For iA = 1 To iAttacker_Initial
      For iD = 0 To iDefender_Initial
         If Absorb_States(iA, iD + 1) > 0 Then
            Range("E28").Offset(i, 0) = iA & " vs " & iD
            Range("E28").Offset(i, 1) = Absorb_States(iA, iD + 1) / _
               (iBatch * iReplications)
            Range("E28").Offset(i, 1).NumberFormat = "0.0000"
            Call Format_Borders(Range("E28").Offset(i, 0), 2)
            Call Format_Lose_States(Range("E28").Offset(i, 0))
```

```
            i = i + 1
          End If
        Next
    Next


Call Animate_Clear

Call View_Sheet("Sim Results", True)
Range("B2").Select
Sim_Run_Menu.Hide
Application.ScreenUpdating = True

MsgBox "Simulation completed " & iReplications * iBatch & " total runs.", , "Finished"


For iA = 1 To iAttacker_Initial
    For iD = 1 To iDefender_Initial + 1
        If Absorb_States(iA, iD) <> 0 Then
            Cells(9, 11 + k) = iA & " vs " & iD - 1
            Cells(10, 11 + k) = Absorb_States(iA, iD) / (iBatch * iReplications)
            k = k + 1
        End If
    Next
Next

End Sub
Function Dice_Roll(iA As Variant, iD As Variant, X As Integer, Y As Integer) As
Variant
Dim j As Integer
Dim iAttacker_Dice(3) As Integer, iDefender_Dice(3) As Integer
Dim iAttacker_Max(2) As Integer, iDefender_Max(2) As Integer
Dim iDefender_Strategy_Sim As Integer
Dim animation As Boolean
Dim datTime As Date
Dim varAnimate(3, 2) As Variant
If Y = 2 Then
    If Rnd() < cProb2 Then
        iDefender_Strategy_Sim = 2
    Else: iDefender_Strategy_Sim = 1
    End If
Else: iDefender_Strategy_Sim = 1
End If

If blnAnimate = True Then
    Call Animate_Blue_Tanks(iAttacker_Strategy(X, Y))
```

```
   Call Animate_Red_Tanks(iDefender_Strategy_Sim)
   Range("B2").Select
   Application.ScreenUpdating = True
   Application.ScreenUpdating = False
   datTime = Timer + 0.5 + (Speed / 10)
   Do
      DoEvents
   Loop Until Timer >= datTime
End If

For j = 1 To 3
   If j <= iAttacker_Strategy(X, Y) Then
      iAttacker_Dice(j) = Int(MRG32k5a * 6 + 1)
      If blnAnimate = True Then Call Animate_Blue_Roll(iAttacker_Dice(j), j)
   Else: iAttacker_Dice(j) = 0
   End If
   If j <= iDefender_Strategy_Sim Then
      iDefender_Dice(j) = Int(MRG32k5a * 6 + 1)
      If blnAnimate = True Then Call Animate_Red_Roll(iDefender_Dice(j), j)
   Else: iDefender_Dice(j) = 0
   End If
Next

If blnAnimate = True Then
   Range("B2").Select
   Application.ScreenUpdating = True
   Application.ScreenUpdating = False
   datTime = Timer + 0.5 + (Speed / 10)
   Do
      DoEvents
   Loop Until Timer >= datTime
End If

iAttacker_Max(1) = Ap.Max(iAttacker_Dice)
iAttacker_Max(2) = Ap.Median(iAttacker_Dice)
iDefender_Max(1) = Ap.Max(iDefender_Dice)
iDefender_Max(2) = Ap.Median(iDefender_Dice)

If iAttacker_Strategy(X, Y) >= 2 And iDefender_Strategy_Sim = 2 Then
   If iAttacker_Max(1) > iDefender_Max(1) And iAttacker_Max(2) > iDefender_Max(2)
Then
      Dice_Roll = Array(iA, iD - 2, 0, 0)
      varAnimate(1, 1) = "W"
      varAnimate(1, 2) = "W"
   ElseIf iAttacker_Max(1) <= iDefender_Max(1) And iAttacker_Max(2) <=
iDefender_Max(2) Then
```

```
      Dice_Roll = Array(iA - 2, iD, 1, 2)
      varAnimate(1, 1) = "L"
      varAnimate(1, 2) = "L"
   Else
      Dice_Roll = Array(iA - 1, iD - 1, 0, 1)
      If iAttacker_Max(1) > iDefender_Max(1) Then
         varAnimate(1, 1) = "W"
         varAnimate(1, 2) = "L"
      Else
         varAnimate(1, 1) = "L"
         varAnimate(1, 2) = "W"
      End If
   End If
Else
   If iAttacker_Max(1) > iDefender_Max(1) Then
      Dice_Roll = Array(iA, iD - 1, 0, 0)
      varAnimate(1, 1) = "W"
      varAnimate(1, 2) = 0
   ElseIf iAttacker_Max(1) <= iDefender_Max(1) Then
      If cProb2 = 0 Then
         Dice_Roll = Array(iA - 1, iD, 1, 1)
      Else: Dice_Roll = Array(iA - 1, iD, dbl_W_w_Ratio, 1)
      End If
      varAnimate(1, 1) = "L"
      varAnimate(1, 2) = 0
   Else: MsgBox "ERROR"
   End If
End If
End Function
```

Appendix IV – Random Number Generator

```
'*************************************************************************
'The following are public constants that are used when MRG32k5a is the random number
generator:
Const norm = 2.32831633968346E-10
Const m1 = 4294949027#
Const m2 = 4294934327#
Const a12 = 1154721#
Const a14 = 1739991#
Const a15n = 1108499#
Const a21 = 1776413#
Const a23 = 865203#
Const a25n = 1641052#

Dim s10 As Double, s11 As Double, s12 As Double, s13 As Double, s14 As Double
Dim s20 As Double, s21 As Double, s22 As Double, s23 As Double, s24 As Double
'*************************************************************************
************************
'This subroutine seeds the MRG32k5a random number generator:
Public Sub RNG_Seed()

   Randomize (Timer)   'seeds the RND() function based on the current timer function

   s10 = 2
   s11 = 3
   s12 = 5
   s13 = 7
   s14 = Int(Rnd * m1) 'the last seed value of the first component is randomly initialized

   s20 = 11
   s21 = 13
   s22 = 17
   s23 = 19
   s24 = Int(Rnd * m2) 'the last seed value of the second component is randomly
initialized

End Sub
'*************************************************************************
************************
'This subroutine is a floating-point implementation in VBA of a 32-bit CMRG of order 5
with 2
'...components based on the article by L'ecuyer, P.(1999). "Good parameters and
implementations
'...for combined multiple recursive random number generators." Operations Research,
Vol 47, No 1,
```

```
'...159-164.  The program was originally written in C, and was translated to VBA by
Jordan Lee.
Public Function MRG32k5a() As Double

    'Declaration of local variables
    Dim k As Long                   'temp variable
    Dim p1 As Double                 'next number in component 1
    Dim p2 As Double                 'next number in component 2

    '-------- Component 1 --------
    p1 = (a12 * s13) - (a15n * s10)
    If p1 > 0# Then p1 = p1 - (a14 * m1)
    p1 = p1 + (a14 * s11)
    k = p1 / m1
    p1 = p1 - k * m1
    If p1 < 0# Then p1 = p1 + m1
    s10 = s11
    s11 = s12
    s12 = s13
    s13 = s14
    s14 = p1

    '-------- Component 2 --------
    p2 = (a21 * s24) - (a25n * s20)
    If p2 > 0# Then p2 = p2 - (a23 * m2)
    p2 = p2 + (a23 * s22)
    k = p2 / m2
    p2 = p2 - (k * m2)
    If p2 < 0# Then p2 = p2 + m2
    s20 = s21
    s21 = s22
    s22 = s23
    s23 = s24
    s24 = p2

    '-------- Combination --------
    If p1 < p2 Then
       MRG32k5a = (p1 - p2 + m1) * norm
    Else
       MRG32k5a = (p1 - p2) * norm
    End If

End Function
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
14-06-2012	Graduate Research Project	Jun 2011 – Jun 2012

4. TITLE AND SUBTITLE
The Comparison of Strategies used in the game of RISK via Markovian Analysis and Monte-Carlo Simulation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S
Major Jordan D. Lee, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/IOA/ENS/12-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
lntentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A; APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

9. ABSTRACT
This paper analyzes strategies of the boardgame RISK using Markov chain analysis and Monte-Carlo simulation in order to compare state-based strategies against sequentially dependent or non-memoryless strategy policies. Previous work had focused on calculating the probability of winning based on using all available engagement strategies and battling until either the attacker is unable to continue engaging the enemy or until the defender is annihilated. This research project applied decision analysis methods to look at alternate strategy policies.
Two primary models were utilized to analyze these strategy policies. First, a computer model was developed that would build a Markov chain with the associated transition probabilities based on an initial set of conditions and a specified set of rolling strategies. Second, a Monte-Carlo simulation was developed that would simulate rolling the dice in order to analyze sequentially dependent strategy policies that cannot be modeled via Markov chains. These strategies were then compared based on the attacker's probability of winning and the expected difference between force strengths at the end of a series of engagements.

15. SUBJECT TERMS
RISK, boardgame, wargame, Markov-Chain, Monte-Carlo simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**Dr. James W. Chrissis**
U	**U**	**U**	**UU**	82	**19b. TELEPHONE NUMBER** *(Include area code)* (937)785-3636

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18